UNIVERSIDAD DE MÁLAGA

E.T.S. INGENIERÍA INFORMÁTICA

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA
GRADUADO EN INGENIERÍA INFORMÁTICA

DESARROLLO DE PANELES DE VISUALIZACIÓN PARA SISTEMAS DE INFORMACIÓN EMPRESARIAL

CONEXIÓN CON BASE DE DATOS EMPRESARIALES INTERNAS

DEVELOPMENT OF VISUALIZATION PANELS FOR BUSINESS INFORMATION SYSTEM

CONNECTION WITH INTERNAL BUSINESS DATABASES

Realizado por
Jose Miguel Castillo Ortiz

Tutorizado por
Francisco López Valverde

Departamento
Lenguajes y Ciencias de la Computación

UNIVERSIDAD DE MÁLAGA

MÁLAGA, JUNIO 2019

Fecha defensa

Fdo. El/la Secretario/a del Tribunal

A mis padres, mi hermano

y todos mis seres queridos que

me han acompañado en este camino.

Gracias.

Resumen

Se ha desarrollado una aplicación web totalmente dinámica mediante el framework Yii2, usando una arquitectura MVC (Modelo/Vista/Controlador), en la cual se elaboran paneles de visualización de datos recogidos desde una base de datos MySQL y gracias a estos paneles se puede ayudar a las empresas en la toma de decisiones ya que se observa la recolección de todos sus datos de una manera más gráfica.

Para la elaboración de estos paneles de visualización se ha realizado un estudio previo de diferentes herramientas gráficas para saber cuál es la que más se adapta a nuestras necesidades y al framework que usamos, este estudio esta detallado en esta memoria y la herramienta utilizada es HighCharts.

Dicho entorno está desarrollado para poder hacer también diferentes tareas como por ejemplo solicitar un registro de alta en la plataforma, su inicio y cierre de sesión, gestión de usuarios, ver los paneles de visualización según tu categoría en la plataforma, entre otras opciones de las cuales entraremos en más profundidad en este documento. Además, todas estas funcionalidades estarán disponibles según el rol de cada usuario ya que se cuenta con diferentes categorías de usuario en la plataforma.

Palabras clave: Yii2, MVC, HighCharts, MySQL, PHP, Business Intelligence, Paneles de visualización.

Abstract

A fully dynamic web application has been developed applying the Yii2 framework, using an MVC architecture (Model/View/Controller), in which panels of data visualization collected from a MySQL database are elaborated. These panels can help the companies in the decision-making process as the collection of all their data can be observed in a more graphic way.

For the elaboration of these visualization panels, a previous study of different graphic tools has been carried out in order to know which one suits best our needs and the framework we use. This study is detailed in this report and the tool used is HighCharts.

This environment is developed to be able to do different tasks as well, such as requesting registration on the platform, starting and stopping the session, managing users, viewing the visualization panels according to your category on the platform, among other options. These options will be analyzed in depth through this document. In addition, all these functionalities will be available according to the role of each user, since there are different user categories in the platform.

Keywords: Yii2, MVC, HighCharts, MySQL, PHP, Business Intelligence, Visualization

Índice

Tabla de ilustraciones

1

Introducción

Hoy en día las empresas de cualquier sector, como por ejemplo el financiero, tecnológico, industrial o empresarial, buscan cual es la mejor forma de tratar todos los datos que tienen almacenados, como son facturas de compra/venta, datos de clientes o zonas de posible expansión según la ubicación de los clientes más potenciales, en definitiva, para poder explotar dichos datos para obtener mayores beneficios y poder minimizar perdidas.

Este trabajo de fin de grado forma parte del área de las tecnologías de la información y se van a exponer las ideas para la creación de una aplicación web relacionada con el sector empresarial.

1.1 Motivación

Mi principal motivación para la elección de este proyecto de fin de grado ha sido por realizar una aplicación web ambientada al sector empresarial, poner en práctica todas aquellas tecnologías aprendidas tanto en el grado como en mi experiencia laboral y poder profundizar en dichas tecnologías para el desarrollo del proyecto como han sido HTML, PHP, CSS.

Además, también mi tutor Francisco López Valverde, profesor de una de las asignaturas que cursé durante el grado, le dio el enfoque que buscaba de poder orientarlo hacia el sector empresarial y crear una herramienta de Business Intelligence, proponiéndome además el framework que uso en la aplicación web, Yii2.

1.2 Objetivos

Uno de los principales objetivos del proyecto es el tratamiento de los datos almacenados en bases de datos internas para a través de ellos crear paneles de visualización sobre ellos. Para poder usar estos paneles se ha tenido que realizar buscando entre las herramientas más demandas hoy en día en este campo para implantarla en el proyecto.

Se busca además que estos paneles sean atractivos y de fácil uso por parte de todos los usuarios de la plataforma.

Otro objetivo ha sido crear una buena interfaz, cómoda y amigable a través del framework Yii2 y ver todas las funcionalidades que se pueden realizar con él, por ejemplo, CRUD de usuarios, donde además se realiza envío de correos electrónicos desde la plataforma, pero todo esto entraremos más en detalle en el capítulo correspondiente.

Por último, se busca que el tratamiento de todos los datos internos en la base de datos, como pueden ser de los usuarios de la plataforma sean tratados de manera correcta.

1.3 Estructura de la memoria

La memoria consta de los siguientes capítulos a partir de aquí:

- **Tecnologías Utilizadas:** Descripción en detalle de cada tecnología usada en el proyecto.

- **Estudio de la herramienta de Business Intelligence:** Estudio realizado para saber la herramienta que se usará en la plataforma web para la creación de los paneles de visualización.

- **Análisis de Requisitos:** Obtención de los requisitos funcionales y no funcionales que van a incluirse en el proyecto.

- **Esquema de diseño:** Elaboración de una maquetación previa a través de la herramienta Balsamiq para seguir cuando se vaya a iniciar la parte de desarrollo de la aplicación.

- **Estructura de la aplicación web:** Explicación en detalle de cada apartado de la aplicación web.

- **Conclusiones y trabajo futuro:** En este último capítulo se valorará el trabajo realizado y posibles mejoras para llevar a cabo en el futuro.

2

Tecnologías Utilizadas

En este segundo capítulo entraremos en detalle sobre las tecnologías utilizadas en el proyecto y el porqué de su uso en él. Estas tecnologías han sido seleccionadas para lograr la máxima optimización del proyecto, es decir que se completasen unas a otras.

2.1 ¿Por qué PHP, HTML, CSS?

Entramos en este primer apartado para profundizar del porqué en el uso de estos tres lenguajes de programación ambientado en web.

En primer lugar, hablaremos sobre PHP *(Hypertext Preprocessor),* el cual ha sido uno de los lenguajes principales para nuestro proyecto junto a HTML debido principalmente a que se ha encargado de dar estructura nuestra aplicación web y darle funcionalidad a esta (Back-end). Aparte de lo mencionado anteriormente también es porque es un lenguaje multiplataforma, porque podemos incrustar código PHP con etiquetas HTML y finalmente porque ofrece conexión con la mayoría de SGBD *(Sistemas de gestión de bases de datos).*

Por otra parte, en HTML *(HyperText Markup Language),* se ha encargado de dar la estructura más orientada al contenido de la web (Front-end), como por ejemplo la creación de formularios (Inicio o registro de usuarios) o la inserción de los paneles de visualización junto a la herramienta que los elabora, la cual entraremos en detalle en el próximo capítulo.

Por último, para dar forma a nuestra aplicación web y para que quede un entorno amigable para el usuario, hemos usado CSS *(Cascading Style Sheets),* que viene a ser hojas de estilos, que viene a ser un mecanismo para describir cómo se va a mostrar por pantalla nuestro entorno.

Una vez planteado los lenguajes de programación para nuestro proyecto (aparte del framework y el lenguaje para la base de datos, los cuales hablaremos a continuación), tocaba ver que entorno desarrollo es el que íbamos a usar para el proyecto.

En principio se empezó con NetBeans, el cual es un entorno de desarrollo integrado libre, pero al cabo de un tiempo dio bastantes problemas a la hora de compilar el proyecto y ejecutarlo, debido a que no compilaba correctamente el proyecto cuando integramos el framework de Yii2 y lo lanzábamos al servidor Apache que nos suministra XAMPP.

Es por ello por lo que se decidió cambiar a Sublime Text, en su versión tercera, ya que es un editor bastante sencillo, pero con lo necesario para no tener problemas a la hora de compilar el proyecto con Yii2 en XAMPP.

2.2 ¿Por qué MySQL con PHPMyAdmin?

Entramos en el apartado de la base de datos y es la que nos va a suministrar tanto los datos para dar forma a los paneles de visualización de nuestra aplicación como los datos internos de la empresa, estos pueden por ejemplo los de los usuarios que tengamos creados y podremos hacer acciones básicas como insertar, actualizar, borrar o seleccionar registros de una forma muy sencilla gracias a MySQL.

Hemos usado este sistema de base de datos ya que es un entorno en el que me siento cómodo trabajando por mi experiencia en él tanto en lo aprendido durante la carrera como en el ámbito laboral que he tenido. Además, XAMPP, nos suministra este SGBD y con el entorno de PHPMyAdmin, el cual cuenta con Licencia Pública General de GNU Versión 2 y que no es necesaria una instalación de este debido a que se maneja desde la web, la parte de base de datos se hace de una manera muy sencilla.

2.3 ¿Por qué Framework Yii2?

En este punto vamos a profundizar sobre el framework que hemos usado en este proyecto y el cual hemos mencionado varias veces en este documento antes.

Antes vamos a responder a que es Yii2 (*Yes It Is*), en primer lugar, es que es un software libre y que además es un framework de PHP de alto rendimiento que con el cual conseguimos crear aplicaciones web modernas en poco tiempo. Es por estos motivos, añadiendo además de que estábamos en la búsqueda de insertar un framework potente en nuestro proyecto y de que a modo personal quería hacerlo con este añadido para poder aprender con frameworks ya que nunca lo había hecho, Yii2 era el software que nos faltaba para poder crear nuestra herramienta.

Tras esta explicación sobre Yii2, a continuación, se exponen varias de las características que nos suministra este framework y hemos usado en nuestro proyecto.

- Patrón de diseño Modelo Vista Controlador (MVC), estilo de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario y la lógica de control. El Modelo contiene una representación de los datos que maneja el sistema, la Vista es la interfaz de usuario y el Controlador actúa como intermediario entre el Modelo y la Vista, es decir, gestionando el flujo de información entre ellos. En la siguiente ilustración se muestra la representación de este patrón de diseño.

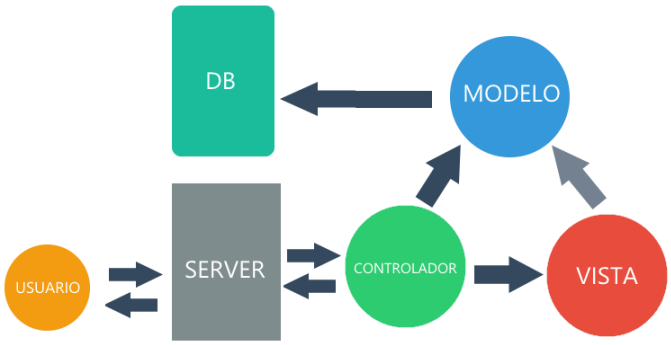

Ilustración 1: Estructura MVC.

- Formularios y validación, mediante expresiones regulares se valida todos los campos de los formularios que se creen.

- Envío de correos electrónicos automatizados, en nuestra herramienta se generan correos una vez se cree un nuevo usuario y esto es posible gracias a Yii2.

Por último, Yii2 no necesita muchos requisitos para su utilización e integración en proyectos y solo es necesario usar PHP 5.4.0 o una versión posterior.

2.4 ¿Por qué XAMPP?

XAMPP es un servidor de plataforma de software libre, el cual principalmente da gestión de bases de datos MySQL, perfecto para nosotros ya que nuestra base de datos es de este tipo, servidor web Apache y los intérpretes para lenguajes de script PHP y Perl, aquí tenemos otro añadido para seleccionarlo por esto último ya que la estructura principal de nuestro proyecto es de PHP.

Otro de los añadidos de usar este servidor es por mi experiencia pasada en su uso para el alojo de páginas web dinámicas, ya que es capaz de hacerlo y satisface todas nuestras necesidades, tanto en páginas creadas a medida con PHP como es el caso de este proyecto, como en el alojo de un CMS (*Content Management System*), como por ejemplo WordPress.

Por último, hay que destacar que incluye un módulo de phpMyAdmin que comentamos anteriormente en el apartado de MySQL y que se encuentra integrado en un panel de administración bastante intuitivo de usar. Este panel nos servirá para arrancar el servidor Apache, la base de datos MySQL, parar estos procesos o configurar parámetros de estos.

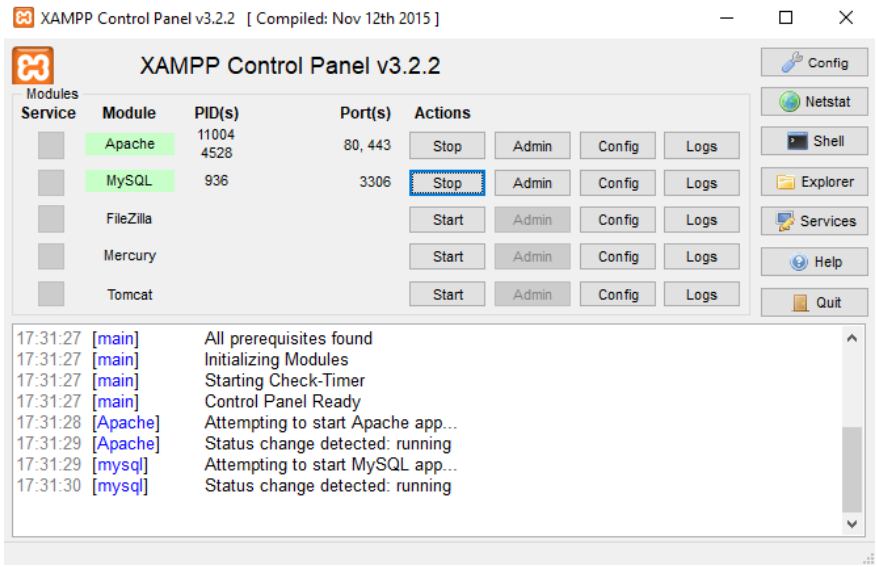

Ilustración 2: Panel de administración XAMPP.

2.5 ¿Por qué GitHub?

GitHub es una plataforma para alojar proyectos utilizando el sistema de control de versiones Git. Esta plataforma está alojada en internet y nos sirve de comunicación entre los desarrolladores del proyecto para ir subiendo los avances realizados o para que el tutor pueda ver el proyecto en cualquier momento hasta que este no esté alojado en internet, pero principalmente es para tener un control de versiones de todo lo que se vaya realizando en el proyecto.

Todo lo anterior se resume en que cuando se terminara una tarea que esté realizada de manera correcta tras probarla, será subida al repositorio de GitHub del proyecto para tener un backup de todo lo que esté bien realizado. Por otra parte, lo he realizado porque son buenas prácticas que no solo se realiza en las empresas que he trabajado sino en todas que realicen proyectos del sector tecnológico.

2.6 ¿Por qué FileZilla?

Una vez que el proyecto tuvo un avance importante, se decidió que se alojara en un servidor suministrado por el tutor. Esto se realizó sobre todo para cuando tuviéramos reuniones no tener que depender de XAMPP o del hardware propio, es decir, cualquier persona desde la dirección alojada podía acceder para ver los avances y para el día de la presentación del proyecto que cualquiera pudiera acceder in situ para poder usar la aplicación web.

Es por ello por lo que se optó por FileZilla, el cual es un software libre que nos permite alojar y compartir archivos en internet, pero principalmente para que un ordenador o dispositivo pueda conectarse a un servidor web.

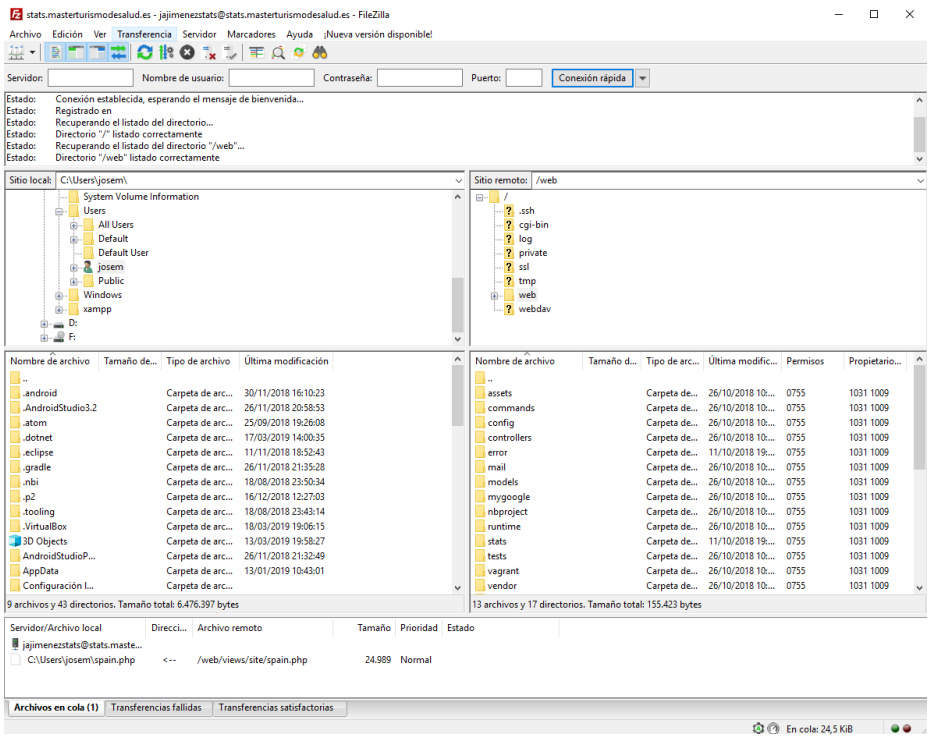

Ilustración 3: Panel de administración FileZilla.

Para terminar con este apartado, la dirección en la que está alojada este proyecto es la siguiente:

http://stats.masterturismodesalud.es/web/index.php

3

Estudio de la herramienta de Business Intelligence

3.1 Introducción

En la actualidad el tema del Business Intelligence está a la orden del día para todo tipo de empresas ya que gracias a estas herramientas transformamos los datos en información y esa información la podemos usar para optimizar el proceso de toma de decisiones de dichas empresas. Para conseguir dicha optimización, lo que buscamos son herramientas que nos permitan crear paneles de visualización con un gran nivel de detalle de los datos proporcionados por parte de las empresas.

Por lo que, en esta primera parte del proyecto, realizaremos un estudio sobre distintas herramientas de Business Intelligence para implementar en nuestra aplicación. En este estudio veremos varias alternativas, de las cuales veremos sus características más destacadas, como pueden ser sus precios de contratación para obtener una licencia, comparándose al final de este apartado entre todas las herramientas que veamos, si es Open Source o gozan de una posibilidad de tenerlo, etc.

Tras esta introducción sobre lo que va a consistir este apartado del proyecto, veremos varias herramientas, las cuales vamos a analizar y estudiar en detalle, para así poder decidir cuál es la que más se adapta a las necesidades del proyecto que vamos a realizar e implementarla en él. Estas herramientas son las siguientes:

- Microsoft Power Bi.
- Tableau Public.
- Visualize.js (JasperReports).
- HighCharts.

Para encontrar estas herramientas, se ha hecho un trabajo previo de investigación para saber cuáles son las herramientas más demandas hoy en día tanto en uso como en satisfacción para los usuarios. Se destacan principalmente entre los usuarios las tres primeras expuestas y las restantes son unas herramientas que hemos encontrado muy interesantes para compararlas con el resto, pero ya las expondremos con detalle en sus puntos correspondiente. Tras esto comenzamos a profundizar en detalle cada una de ellas.

3.2 Microsoft Power BI

Como cabría de esperar Microsoft es una de las empresas que también se ha establecido en el mercado del Business Intelligence ofreciendo a las empresas esta herramienta que vamos a detallar. Power Bi nace con la idea de monitorizar todo lo que concierne a la empresa de una forma más gráfica, la cual nos ofrece un servicio en línea para la creación de paneles a través de los datos que les proporcionemos.

Tras esta breve explicación sobre Power Bi, antes de entrar en el apartado de la contratación, vamos a ver las características más importantes que nos ofrecen para poder usar en nuestro proyecto.

Una de las principales características es que existe la posibilidad de generar código embebido de los paneles de visualización que creemos en la herramienta para poder insertarlo en nuestro proyecto. A continuación, se muestra un ejemplo de dicha posibilidad y también se ha hecho para ir realizando pruebas para una posible implementación futura en el proyecto en caso de ser la herramienta seleccionada.

El ejemplo consiste en que hemos creado unos datos de unos alumnos con sus respectivas notas y los mostramos mediante un panel de visualización de barras y este es el resultado.

```
<div class="jumbotron">
    <iframe width="800" height="600" src="https://app.powerbi.com/view?r=eyJrIjoiODU1YmFhMjYtZmEyZC00MDMwLThmODAtY2FhOTQwMD
            kyZjgyIiwidCI6ImU3ZjUzZjNmLTYzNmItNDNhZC04MDdlLTU3Yzk2NmZmN2RiOCIsImMiOjh9"
            frameborder="0" allowFullScreen="true">

    </iframe>
```

Ilustración 4: Código embebido generado por Power Bi Desktop.

Ilustración 5: Panel de visualización en nuestra aplicación con Power BI (fase de pruebas de herramientas).

Como se ve en el ejemplo, lo que consiste principalmente es desde Power Bi insertar los datos, los cuales explicaremos a continuación como se insertan y que datos acepta la herramienta, generar los paneles de visualización como uno más desee y una vez hecho esto la posibilidad de generar el código embebido del panel creado.

La segunda característica que mencionar es sobre los datos, Power Bi acepta una gran cantidad de datos para su herramienta, donde las más principales son CSV, Excel, Google Analytics, MailChimp, MySQL (destacar esta posibilidad porque la base de datos de nuestro proyecto está creada en MySQL), etc.

No todo va a ser ventajas en esa herramienta, al realizar las pruebas hay que destacar un factor en contra que es que al generar los paneles de visualización e insertar el código embebido, en el caso de que exista la necesidad de actualizar los datos, habría que realizar el proceso desde cero, es decir, crear los paneles de visualización desde los nuevos datos actualizados y generar su nuevo código embebido para su inserción por lo que no es óptimo.

Por último, antes de entrar al aspecto de contrataciones, hay que destacar que desde hace unos años Microsoft abrió un repositorio en GitHub para que la comunidad, el cual es el siguiente:
https://github.com/Microsoft/PowerBI-visuals
Vayan subiendo mejoras o arreglos para poder implementarse en futuras actualizaciones o de cara a un uso no comercial por parte del usuario.

Una vez expuesto todas las ventajas y desventajas que nos ofrece la herramienta sobre nuestro proyecto, pasemos al aspecto económico. Power Bi se mueve por tres tipos de contratación, el primero que es el que hemos estado usando es el llamado Power Bi Desktop que con el cual tenemos una capacidad de almacenamiento de hasta 1 Gb de datos y todo lo expuesto anteriormente. Por otra parte, tenemos la versión Pro por un precio de 8.40€ por usuario por mes, donde aquí nos ofrecen la posibilidad aparte de lo anterior de mantener los datos actualizados automáticamente, incluyendo si es el origen es desde almacenamiento local, por lo que solventaría el problema planteado

anteriormente. Y por último cuenta con una versión Pro-Premium, la cual se establece su precio por nodo y por mes, pero es una solución para empresas de gran tamaño.

3.3 Tableau Public

Al igual que Microsoft, tenemos que Tableau tiene una herramienta muy potente para poder realizar Business Intelligence, que es Tableau Public. Tableau Public es una herramienta gratuita, es decir sin costes y la cual inicialmente nos da 10 Gb de almacenamiento de datos.

Las prestaciones que nos da esta herramienta son muy similares a la anterior, pero mejora significativamente la forma de ingresar los datos y también nos da la posibilidad de muchos más formatos entre los que podemos destacar PDFs, archivos de texto, JSON, e incluso la posibilidad de conectarnos a un conector de datos web. Pero, no es posible la conexión a bases de datos de tipo MySQL, un factor en contra a destacar.

Aun así, se aprecian mejoras respecto a Power BI ya que hay mayor diversidad de paneles de visualización a desarrollar, es decir, se tiene un mayor abanico de posibilidades como se puede ver a continuación en el ejemplo que realizamos en el punto anterior.

Podemos obtener el código embebido como antes e insertarlo en nuestra aplicación para poder visualizarlo, quedaría de la siguiente manera.

```
<div class='tableauPlaceholder' id='viz1524158136144' style='position: relative'>
    <noscript>
    <a href='#'><img alt='Dashboard 2 ' src='https:&#47;&#47;public.tableau.com&#47;static&#47;images&#47;Li&#47;Libro2_632&#47;Dashboard2&#47;1_rss.png' style='border: none' /></a>
    </noscript>

    <object class='tableauViz'  style='display:none;'>
        <param name='host_url' value='https%3A%2F%2Fpublic.tableau.com%2F' />
        <param name='embed_code_version' value='3' />
        <param name='site_root' value='' />
        <param name='name' value='Libro2_632&#47;Dashboard2' />
        <param name='tabs' value='no' />
        <param name='toolbar' value='yes' />
        <param name='static_image' value='https:&#47;&#47;public.tableau.com&#47;static&#47;images&#47;Li&#47;Libro2_632&#47;Dashboard2&#47;1.png' />
        <param name='animate_transition' value='yes' />
        <param name='display_static_image' value='yes' />
        <param name='display_spinner' value='yes' />
        <param name='display_overlay' value='yes' />
        <param name='display_count' value='yes' />
        <param name='filter' value='publish=yes' />
    </object>
</div>

<script type='text/javascript'>
    var divElement = document.getElementById('viz1524158136144');
    var vizElement = divElement.getElementsByTagName('object')[0];
    if ( divElement.offsetWidth > 800 ) {
        vizElement.style.width='1080px';
        vizElement.style.minHeight='747px';
        vizElement.style.maxHeight='887px';
        vizElement.style.height=(divElement.offsetWidth*0.75)+'px';
    } else if ( divElement.offsetWidth > 500 ) {
        vizElement.style.width='1080px';
        vizElement.style.minHeight='747px';
        vizElement.style.maxHeight='887px';
        vizElement.style.height=(divElement.offsetWidth*0.75)+'px';
    } else {
        vizElement.style.width='1080px';
        vizElement.style.minHeight='747px';
        vizElement.style.maxHeight='887px';
        vizElement.style.height=(divElement.offsetWidth*1.77)+'px';
    }
    var scriptElement = document.createElement('script');
    scriptElement.src = 'https://public.tableau.com/javascripts/api/viz_v1.js';
    vizElement.parentNode.insertBefore(scriptElement, vizElement);
</script>
```

Ilustración 6: Código embebido generado por Tableau Public.

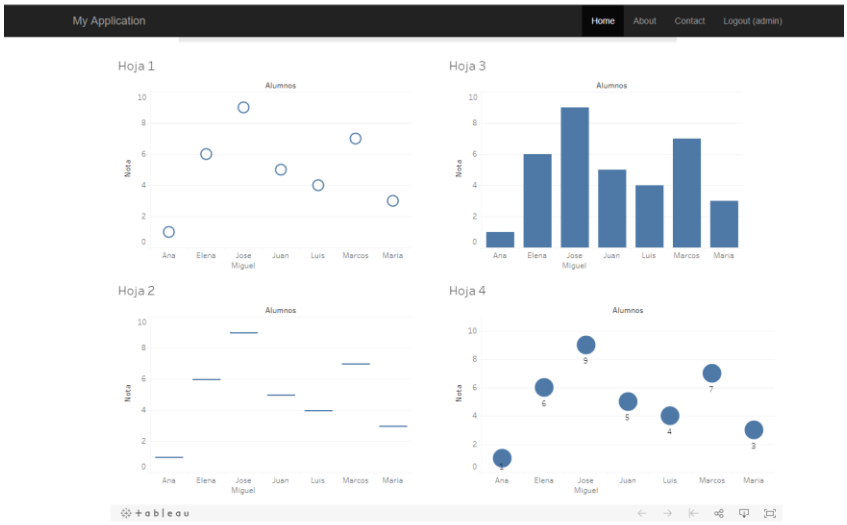

Ilustración 7: Panel de visualización en nuestra aplicación con Tableau Public (fase de pruebas de herramientas)

Tras realizar estas pruebas, como ya dijimos anteriormente se ven mejoras respecto a Microsoft Power BI, además los cambios que tengan los datos almacenados, en esta herramienta se actualizarán en la próxima recarga de la aplicación, por lo que es un punto a favor respecto a la anterior.

3.4 Visualize.js (JasperReports)

JasperReports es una de las grandes herramientas en el desarrollo de gestión de reportes para las empresas y es por ello por lo que se han ido expandiendo más por los distintos productos que ofrecen, los cuales destacamos Visualize.js como herramienta para BI que es la que vamos a exponer en este apartado.

Jaspersoft, que es la empresa que proviene tanto JasperReports como Visalize.js, es totalmente Open Source, pero tenemos dos ediciones distintas la de comunidad y las de empresa, pero la parte de Visualize.js solo está incluida durante un periodo de tiempo corto, unos 60 días y si se desea tener se tiene que tener la de empresa, la cual tiene costo y aun así también hay comunidad aportando en ella. Para saber el costo a pagar hay que contactar con la empresa Jaspersoft, la cual no me ha respondido, pero investigando normalmente es por horas de uso de la aplicación que suele ser de 1€ la hora, pasándolo de dólares a euros.

Cuenta con todas las posibilidades expuestas en los puntos anteriores, aunque la forma de ingresar de datos no sea tan extensa como las anteriores, tiene la posibilidad de conectarte a tu base de datos como puede ser de MySQL, como es nuestro caso.

Por lo que aquí mostramos otro ejemplo de esta herramienta como hemos realizado en los casos anteriores para poder ver su posible implementación en el proyecto. Su código embebido es insertar desde el HTML el script que llama al servidor de JasperReports

para ejecutar visualize.js, donde se especifica las credenciales de acceso y la función a realizar.

```
<script src="https://mobiledemo.jaspersoft.com/jasperserver-pro/client/visualize.js"></script>

<div id="container"></div>
```

Ilustración 8: Script para llamar a Visualize.js

```
visualize({
    auth: {
        name: "joeuser",
        password: "joeuser",
        organization: "organization_1"
    }
}, function (v) {

    //render report from provided resource
    v("#container").report({
        resource: "/public/Samples/Reports/States",
        error: handleError
    });

    //show error
    function handleError(err) {
        alert(err.message);
    }

});
```

Ilustración 9: Credenciales para acceder (usuario y contraseña) y función a realizar.

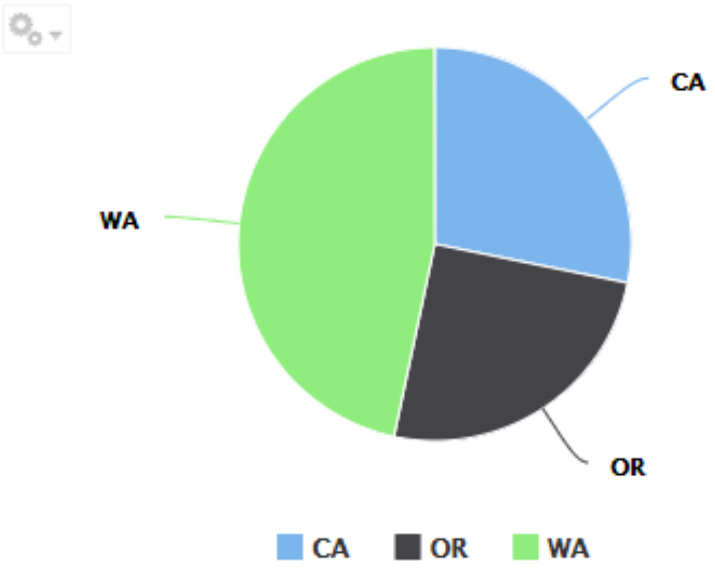

Ilustración 10: Paneles de visualización resultado.

3.5 HighCharts

HighCharts es una empresa que se dedica exclusivamente a la creación de paneles de visualización para que las empresas los puedan usar para explotar sus datos de una forma muy visual, ya que, al estar enfocados solo al aspecto de visualización, estos son de gran calidad y detallados.

HighCharts es muy conocida ya que trabaja con empresas como son Facebook, Twitter, Yahoo!, Visa, etc., es decir, un gran abanico de grandes empresas mundiales.

Las características principales de HighCharts son que al tener los módulos que contienen los paneles de visualización, estos se añaden al proyecto y ya se pueden incluir sin ningún tipo de programa externo, por lo que es mucho más sencillo que el resto de las opciones expuestas con anterioridad. Por otra parte, el apartado de integridad de datos, como ya hemos dicho que los módulos están en el proyecto integrados, se pueden obtener desde él de cualquier forma, incluyendo una base de datos externa como puede ser en nuestro caso de MySQL.

Para poder contratar HighCharts se tiene que comprar cada módulo y cada uno de ellos tiene duración permanente, es decir, una vez que se tenga lo tendrás para siempre desde la versión que la contratas junto con las anteriores, pero en el caso de futuras versiones nuevas, se debería comprar como un nuevo módulo. Para el proyecto que estamos realizando vamos a tener en cuenta tres de los seis módulos que ofertan, ya que los tres que mencionamos están dirigidos al tema de paneles de visualización.

Los precios que ofertan para cada módulo y partiendo que es para un solo desarrollador, ya que ofertan dentro de estos paquetes según cuantos desarrolladores se tengan, son los siguientes:

- HighCharts JS: 359€
- HighStock JS: 711€
- HighMaps JS: 359€

Aunque también da la opción de tener los 3 módulos por el precio de 1200€.

Una vez sabido todo lo referente a sus características y precios de contratación, como hemos realizado en los casos anteriores, vamos a crear una prueba para saber cómo quedaría en nuestro proyecto de cara a una posible implementación. En este caso usaremos la versión no comercial que nos suministra HighCharts, que se puede usar para uso personal para cualquier persona o para una empresa en el ámbito de gestiones internas, es decir en cuanto se quiera usar para temas de beneficio económico no se podría usar este tipo de licencia.

Para realizar las pruebas incluiremos los módulos de HighCharts en nuestro proyecto de prueba e incluiremos, algunos paneles de visualización que tenemos en el módulo de HighCharts JS. El resultado obtenido es el siguiente:

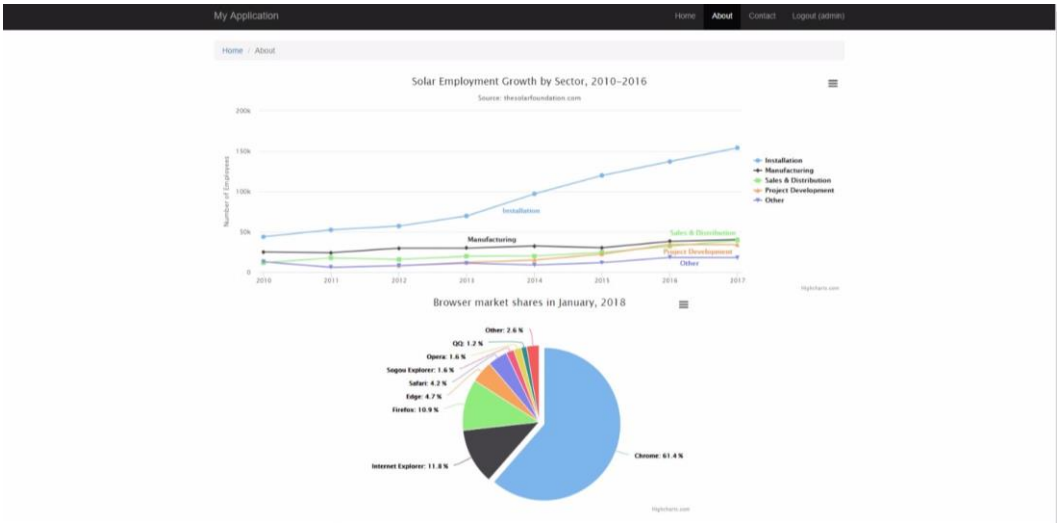

Ilustración 11: Prueba de paneles de visualización de HighCharts.

3.6 Valoraciones finales y elección de herramienta

Tras lo expuesto anteriormente, llegamos a la parte final de este estudio, en esta parte compararemos en una tabla cada herramienta para poder llegar a una elección definitiva.

En dicha tabla veremos cuatro aspectos que deben cumplir la herramienta para poder ser usada en nuestro proyecto, primero que el código sea embebido, por otro lado, la posibilidad de una conexión a una base de datos MySQL, en relación con este aspecto tenemos que los datos cambiados en dicha base de datos se actualicen los paneles sin tener que tocar nada del código embebido y por último valoraremos el precio de coste de la herramienta, este aspecto valoraremos el precio de costo de un año y en el apartado de HighCharts que vimos que tenían varios módulos usaremos la no comercialización principalmente, pero estableceremos un precio inicial de compra del pack de los tres módulos que mencionamos en su apartado anterior.

	Código embebido	Conexión BD MySQL	Actualización de datos sin cambiar código	Precio
Power Bi				100€ (usuario/año)
Tableau Public				Gratuito
Visualize.js				1€/hora
HighCharts				1200€ (una sola vez)

Como podemos observar de la tabla anterior sacamos la conclusión que las dos herramientas que cumplen principalmente las necesidades a cubrir para el proyecto son Visualize.js y HighCharts (sin contar el precio). A partir de aquí, debemos de valorar el apartado económico por parte de la empresa que es lo que más rentable que le sale de cara al futuro si pagar por módulos que necesite como ofrece HighCharts o lo que ofrece Visualize.js que es el pago por hora de uso, en este caso la opción más viable es la que nos ofrece HighCharts y es por el siguiente motivo.

Como el proyecto principalmente se va a enfocar a un apartado no comercial, como ya dijimos HighCharts nos ofrece dicha posibilidad, en cambio Visualize.js no nos la ofrece, por lo que si de cara a un futuro vemos la necesidad de obtener un beneficio económico por el proyecto que vamos a desarrollar lo único que se deberá tener en valoración será cuando módulos comprar.

En conclusión, tras todo lo expuesto en este apartado del proyecto, vamos a integrar la herramienta de HighCharts para el modelado de los paneles de visualización que necesitamos en ella.

4

Análisis de Requisitos

En este capitulo veremos el análisis de requisitos tanto funcionales como no funcionales que contendrá nuestro proyecto, este es el paso final antes de empezar a programar.

4.1 Requisitos Funcionales

	Requisitos	Descripción
RF-01	CRUD Usuarios	- RF-01-1 Create: mediante un formulario de registro, en el cual se rellenarán campos tanto obligatorios como no obligatorios para la creación de un usuario (aquí entra en función el RF-06). - RF-01-2 Read: desde el apartado de perfil, el usuario podrá ver toda su información. - RF-01-3 Update: desde el mismo apartado de perfil, se tendrá la posibilidad de modificar datos erróneos o nuevas actualizaciones a su perfil. - RF-01-4 Delete: el usuario podrá eliminar su cuenta en cualquier momento.

RF-02	CRUD Paneles de visualización	- RF-02-1 Create: cada usuario podrá crear sus propios paneles de visualización y tener todos los que desee, cada uno se podrá personalizar. - RF-02-2 Read: tras crear un panel de visualización, podrá verlo inmediatamente, en cambio, existirá una zona en el sitio web donde ver todos los paneles creados por un usuario. - RF-02-3 Update: tras crear un panel de visualización, este será totalmente modificable, es decir, se podrá cambiar el nombre del panel, la cantidad de gráficos a mostrar o los datos que aparezcan en ellos. - RF-02-4 Delete: el usuario podrá eliminar paneles de visualización que ya no vea conveniente usar.

RF-03	Iniciar sesión	Tras completar el registro en el sitio web, el usuario iniciará sesión para poder usar todas las funciones que contiene.

RF-04	Cerrar sesión	Si el usuario ve conveniente cerrar su sesión, tendrá dicha posibilidad.

RF-05	Validación de datos	A la hora de rellenar formularios como pueden ser de registro, de modificaciones por ejemplo en el perfil o de crear paneles de visualización, se validarán los datos insertados para ver si son correctos y en caso de fallo se le notificará al usuario para su corrección.

RF-06	Modales de confirmación/eliminar	A la hora de crear un usuario, panel de visualización, etc., se visualizará un modal de confirmación para que el usuario de su consentimiento de aceptación, así mismo, será para la eliminación.

RF-07	Histórico de paneles de visualización	Apartado donde un usuario podrá visualizar todos los paneles de visualización que haya creado.

RF-08	Buscador en la administración de usuarios	Cuando el administrador necesite buscar un usuario en concreto tendrá la posibilidad de hacerlo más rápido mediante un buscador.

RF-9	Contacto	Posibilidad de que el usuario o una empresa interesada o con cualquier duda, contacte con el responsable mediante un apartado de contacto donde se le enviará un correo electrónico.

4.2 Requisitos No Funcionales

	Requisitos	**Descripción**
RNF-01	Tiempos de carga	La aplicación no tendrá tiempos de carga excesivos y será muy fluida cuando se use.

RNF-02	Datos modificados	Al modificar datos, estos se deberán asegurar que se insertan los datos correctamente.

RNF-03	Actualización de la Base de Datos	Cuando haya modificaciones, estas serán modificadas inmediatamente en la base de datos.

RNF-04	Seguridad en datos críticos	Datos críticos como pueden ser datos de la empresa o las contraseñas de los usuarios que estén registrados, serán cifrados para mayor seguridad.

RNF-05	Diseño “Responsive”	Adaptabilidad a cualquier tipo de pantalla.

RNF-06	Interfaces gráficos	Todos los paneles de visualización que se creen por parte de un usuario estarán optimizados para que no haya fallos.

RNF-07	Disponibilidad	La aplicación tendrá disponibilidad total, es decir, 24/7 los 365 días del año.

RNF-08	¿Compatibilidad?	La aplicación se podrá usar en los navegadores de más uso como son Google Chrome o Firefox.

RNF-09	Validar formularios	Al rellenar un formulario de registro/crear paneles, los datos que sean obligatorios se le notificará al usuario para su relleno.

5

Esquema de diseño

5.1 ¿Por qué Balsamiq?

A la hora de crear una nueva aplicación y más cuando es orientada al entorno web es necesario plasmar bien en un papel el esquema y funcionalidades que van a tener, por lo que de ahí nace la idea de usar Balsamiq para poder crear nuestros bocetos de como va a ser nuestro entorno web estructurado.

Esta aplicación nos facilita mucho este trabajo ya que podemos crear todo nuestro proyecto con su navegación en diferentes entornos como pueden ser en escritorio, móvil o Tablet.

Antes de mostrar algunas imágenes del boceto que nos sirvió para orientarnos, decir que es un software de pago, pero que cuenta con una versión escritorio gratuita con la mayoría de las funcionalidades necesarias para crear el boceto.

Tras acabar de realizar esta parte, se da por concluida toda la parte de búsqueda de herramientas, estudio de los paneles de visualización a usar y análisis de requisitos y comenzaremos con el desarrollo del proyecto.

5.2 Muestra del diseño

No se va a mostrar todo el boceto del proyecto ya que sería cargar muchas imágenes el documento, pero sí alguna de las partes más destacadas. Este fue el boceto creado, pero luego a la hora de desarrollar surgen ideas y puede tener ciertas modificaciones la interfaz respecto a este modelo.

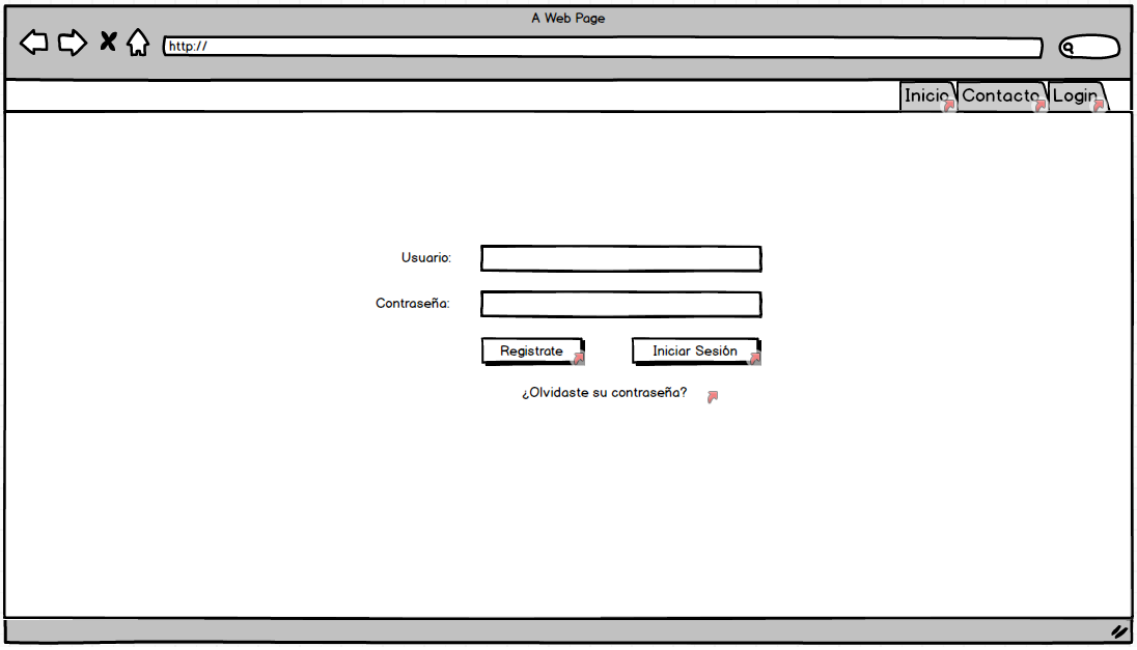

Ilustración 12: Boceto Inicio de Sesión.

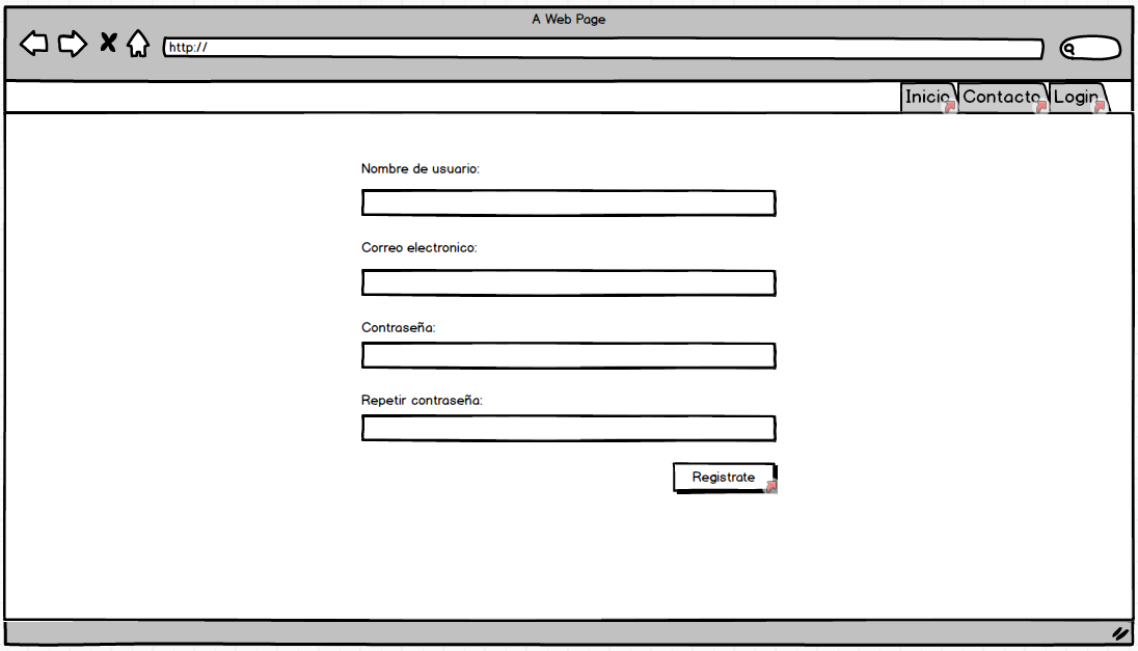

Ilustración 13: Boceto Registro de Usuario.

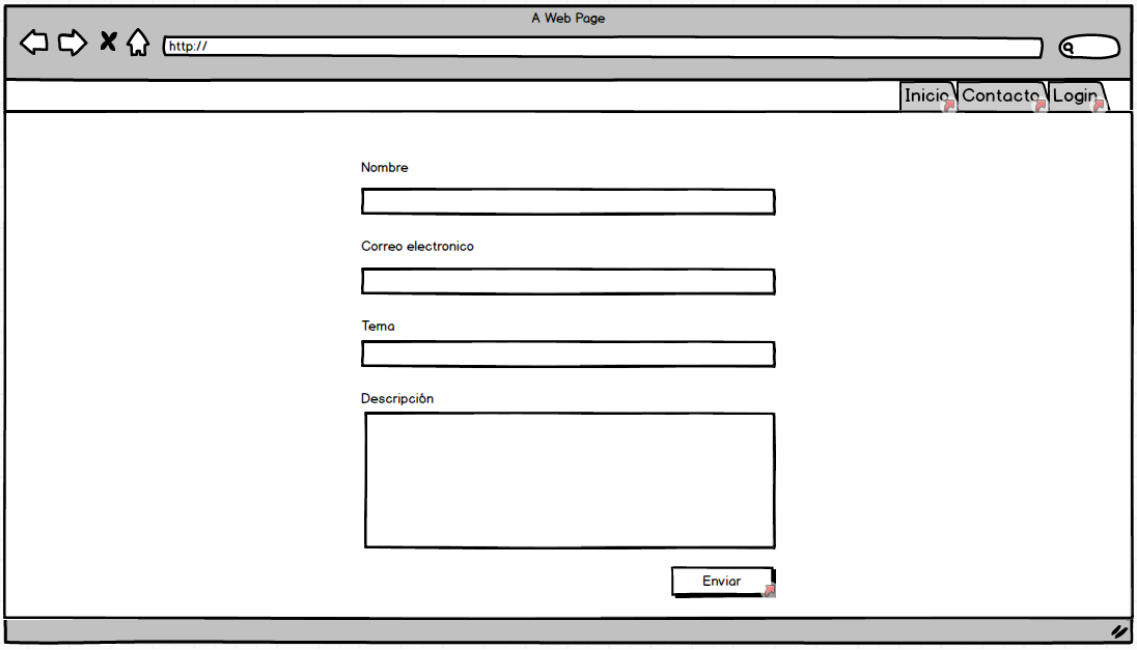

Ilustración 14: Boceto Contacto al Administrador.

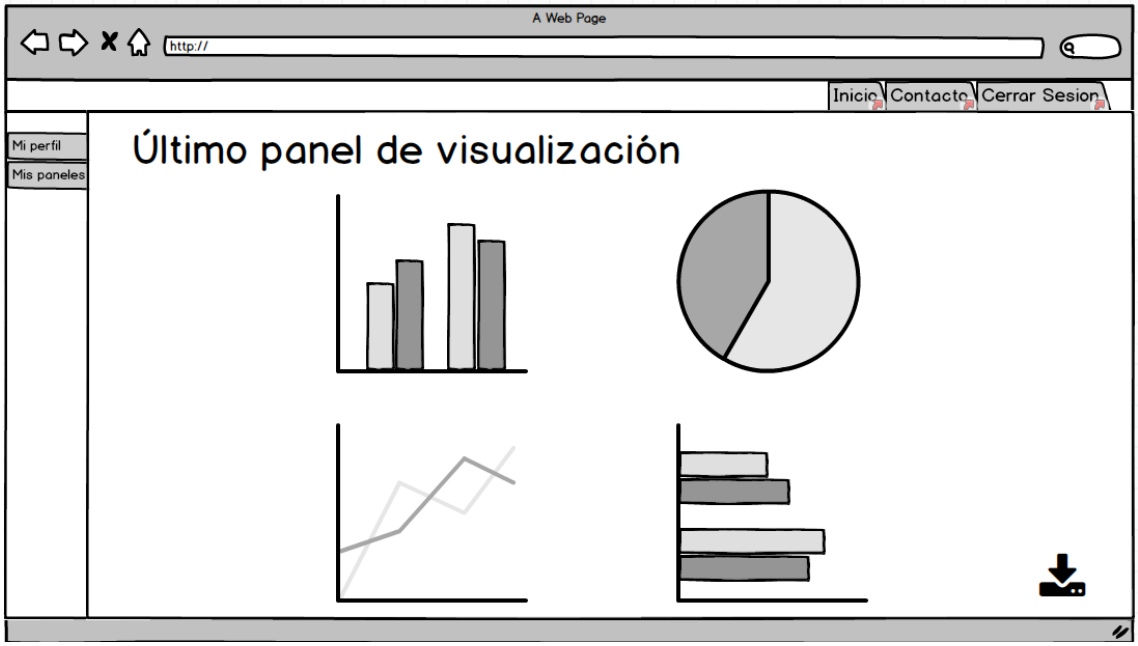

Ilustración 15: Boceto "Mis paneles".

6

Desarrollo de la aplicación

6.1 Introducción

Una vez realizado todo lo expuesto en los capítulos anteriores, toca ponerlos en práctica para la creación de nuestra aplicación web, pero antes de comenzar a explicar como se ha realizado, vamos a exponer cual ha sido la temática que tiene dicha aplicación.

La aplicación consiste en tener una aplicación donde distintos usuarios de una empresa puedan entrar para poder ver sus paneles de visualización según el rol de departamento que tenga asignado, que en este caso los departamentos están especificado por países, por ejemplo, España, Inglaterra, etc. y además tendrán el apartado de su perfil para poder modificar sus datos personales.

Por otra parte, también tendremos un rol de administrador que podrá ver todos los paneles de visualización, entre otras funcionalidades orientadas a este rol, como son la administración de los usuarios de la aplicación como así los registros de nuevos usuarios que se vayan recibiendo.

En los próximos subpuntos de este capítulo iremos extendiendo todo lo explicado en los párrafos anteriores.

6.2 Inicio sin iniciar sesión

Cuando cualquier usuario tanto que este registrado como no, se encuentra la siguiente interfaz.

Ilustración 16: Interfaz principal sin iniciar sesión.

Como se puede observar se ven unos ejemplos de como son los paneles de visualización realizados mediante la herramienta de HighCharts.
Por último, en el margen superior derecho, vemos las dos funcionalidades que podemos realizar y estarán su explicación se expondrá en el siguiente subcapítulo.

6.3 Iniciar Sesión y Solicitar registro

Vamos a exponer en primer lugar el inicio de sesión de los usuarios (apartado 1 en rojo de la ilustración 16).

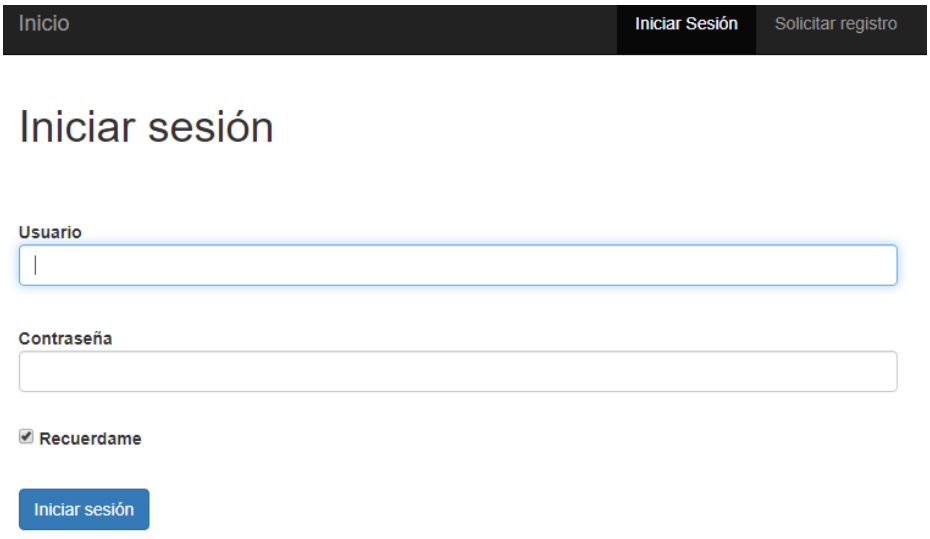

Ilustración 17: Interfaz iniciar sesión.

Este apartado fue uno de los primeros que realicé con Yii2 y la estructura de MVC. La vista la tenemos implementada con código PHP y HTML como se muestra en la ilustración anterior y para enlazarla con el modelo cogemos los parámetros introducidos en las cajas de texto.

```php
<?= $form->field($model, 'usuario')->textInput(['autofocus' => true]) ?>

<?= $form->field($model, 'contraseña')->passwordInput() ?>
```

Ilustración 18: Código PHP parámetros de inicio de sesión.

Estos parámetros quedan almacenados en unas variables para comprobar en primer lugar que cumplen los requerimientos de que los campos no estén vacíos, que existan los usuarios introducidos y por otro lado que la contraseña sea válida. Tras esto se lanza al controlador para ejecutar la acción si todo es correcto.

```php
public function rules()
{
    return [
        // username and password are both required
        ['usuario', 'required', 'message' => 'Introduzca nombre de usuario'],
        ['contraseña', 'required', 'message' => 'Introduzca contraseña'],
        // rememberMe must be a boolean value
        ['recuerdame', 'boolean'],
        // password is validated by validatePassword()
        ['contraseña', 'validatePassword'],
    ];
}

public function validatePassword($attribute, $params)
{
    if (!$this->hasErrors()) {
        $user = $this->getUser();

        if (!$user || !$user->validatePassword($this->contraseña)) {
            $this->addError($attribute, 'Contraseña incorrecta.');
        }
    }
}

public function login()
{
    if ($this->validate()) {
        return Yii::$app->user->login($this->getUser(), $this->recuerdame ? 3600*24*30 : 0);
    }
    return false;
}

public function getUser()
{
    if ($this->_user === false) {
        $this->_user = User::findBynombre($this->usuario);
    }

    return $this->_user;
}
```

Ilustración 19: Código modelo inicio de sesión.

En esta ilustración podemos ver lo explicado anteriormente de como se realiza la comprobación de que los campos estén correctos mediante las funciones rules y validatePassword y mediante getUser obtenemos si dicho usuario existe en la base de datos para su inicio de sesión.

Por otro lado, tenemos el apartado de Solicitar Registro (apartado 2 en rojo de la ilustración 16), pero este tiene una peculiaridad que consiste en que las solicitudes de registro a la plataforma son vía email, es decir, todas las solicitudes que se hagan llegan a un correo electrónico del administrador del sistema y este ya realiza las altas desde su apartado en la aplicación. Esto se ha hecho de esta forma porque Yii2 lo permite y es algo que me gustaba que estuviera en el proyecto ya que lo aprendí cuando empecé a probar Yii2.

Ilustración 20: Interfaz Solicitar Registro.

Como se puede observar sigue una estructura como la de inicio de sesión y tenemos el añadido de tener un código de verificación para que no se haga solicitudes masivas. Además, como se puede ver todo sale en un color verdoso y es que todos los campos introducidos están correctos, se comprueba en tiempo real como en el inicio de sesión con la función rules. Tras esto enviamos la solicitud y vemos nuestro correo electrónico para ver el mensaje recibido.

Ilustración 21: Solicitud de alta mediante email.

Para poder realizar este proceso de conexión entre nuestra plataforma y el correo electrónico, hay que crear este proceso en el archivo de configuración que tiene Yii2 y es como se muestra a continuación.

```
$config = [
    'id' => 'basic',
    'basePath' => dirname(__DIR__),
    'bootstrap' => ['log'],
    'aliases' => [
        '@bower' => '@vendor/bower-asset',
        '@npm'   => '@vendor/npm-asset',
    ],
    'components' => [
        'request' => [
            // !!! insert a secret key in the following (if it is empty) - t
            'cookieValidationKey' => '-VR4uk_yPfdfvjOvfCE0imSpNE9ZuC-h',
        ],
        'cache' => [
            'class' => 'yii\caching\FileCache',
        ],
        'user' => [
            'identityClass' => 'app\models\User',
            'enableAutoLogin' => true,
        ],
        'errorHandler' => [
            'errorAction' => 'site/error',
        ],
        'mailer' => [
            'class' => 'yii\swiftmailer\Mailer',
            'useFileTransport' => false,
            // send all mails to a file by default. You have to set
            // 'useFileTransport' to false and configure a transport
            // for the mailer to send real emails.
            'transport' => [
            'class' => 'Swift_SmtpTransport',
            'host' => 'smtp.gmail.com',
            'username' => 'tfg2018josemi@gmail.com',
            'password' => '790348925',
            'port' => '587',
            'encryption' => 'tls',
```

Ilustración 22: Código conexión plataforma/Email.

6.4 Pantalla principal administrador

Una vez expuesto varias de las partes comunes, es decir, independientemente del tipo de usuario que sea uno, vamos a explicar como es la aplicación web según el administrador de todo el sistema.

Cuando inicia sesión el usuario administrador, este se va a encontrar de primeras en la página de inicio un panel con gráficos sobre datos de un departamento que tenga asignado, en este caso es el de España.

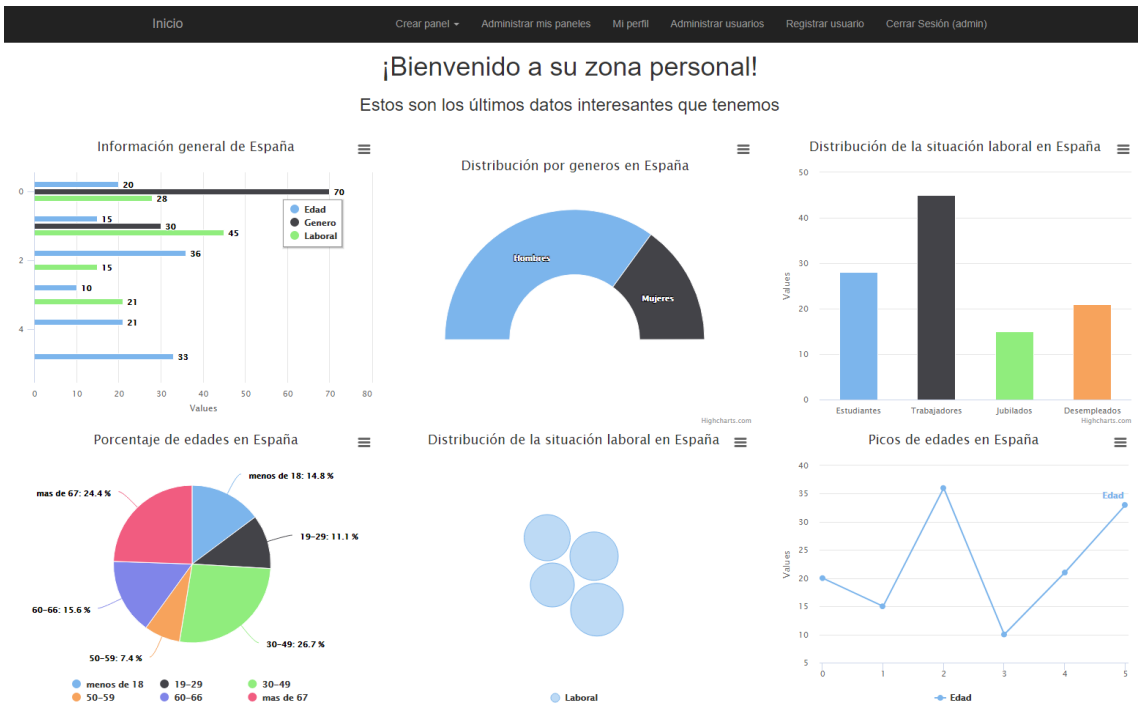

Ilustración 23: Pantalla de inicio administrador.

Para poder realizar este cambio en el inicio respecto al apartado anterior, comprobamos desde el controlador que tipo de usuario tenemos, es decir si está registrado o no y este proceso se realiza con la siguiente acción.

```php
public function actionIndex()
{
    if (!Yii::$app->user->isGuest) {
        return $this->render("indexlogueado");
    }
    return $this->render('index');
}
```

Ilustración 24: Acción cambio de índice en el controlador.

6.4.1 Crear panel

Uno de los aspectos más importantes en este proyecto es la posibilidad de que cada usuario pueda crear sus propios paneles de visualización con los datos que tenga disponible, guardarlos para de cara a un futuro poder mostrarlos en otro momento, editarlos, es decir, poder cambiar los datos de un gráfico, cambiar el tipo de gráfico, el número de gráficos que tiene ese panel de visualización o incluso eliminar dicho panel si ya no es necesario.

En este punto vamos a explicar como ha sido el proceso por el cual podemos crear los paneles de visualización y la parte de poder editarlos y borrarlos, lo explicaremos en el siguiente apartado que está dedicado a la administración de los paneles que tenga cada usuario.

Ilustración 25: Creación de un panel de visualización.

Como se puede ver tenemos en primer lugar un cuadro de mandos, donde podemos ocultarlo o mostrarlo para tener una mayor visión del panel que estamos creando, por otra parte, en el botón de configurar paneles podemos decidir cuantos gráficos va a tener nuestro panel, desde un único gráfico hasta un máximo de seis y por último si decidiéramos guardar dicho panel deberemos indicarle un nombre.

Para poder realizar este proceso mediante el MVC vamos a explicar cada uno de ellos a continuación.

La vista (V) que vemos en la anterior imagen ha sido creada mediante la combinación de los lenguajes PHP, HTML y JavaScript para las acciones y con CSS para darle un estilo más personal, con SQL para modelar los datos en los gráficos y estos gráficos creados mediante HighCharts.

```
<head>
    <h1 style = "text-align: center">Panel de información sobre España</h1>

    <div style = "text-align: center">
        <h3><?= $msg ?></h3>
        <?= Html::button("Configurar paneles", array('class ' => "btn btn-primary", 'onclick' => 'js:openNav()')); ?>
        <?= Html::button("Mostrar Cuadro de Mandos", array('class ' => "btn btn-primary", 'onclick' => 'js:mostrarCuadro()'));
        ?>
        <?= Html::button("Ocultar Cuadro de Mandos", array('class ' => "btn btn-primary", 'onclick' => 'js:ocultarCuadro()'));
        ?><br></br>
        <?= $form->field($model, "numGraficos")->input("hidden", ["id" => 'numGraficos'])->label(false) ?>
    </div>

</head>
```

Ilustración 26: Cuadro de mandos de Crear Panel.

En esta ilustración tenemos el código usado para las acciones de los botones que vemos debajo del título del panel con sus correspondientes acciones en JavaScript, además tenemos una "Field" oculta para poder enviar a nuestro modelo el número de gráficos que va a tener nuestro panel.

Por otra parte, en el cuadro de mandos tenemos para elegir que tipos de gráficos y que datos se van a mostrar en cada uno de los que hayamos decidido poner.

```
    <div id="mando5" class="col-sm-2 col-xs-6">
        <?=
        $form->field($model, 'tipoGrafico5')->dropDownList(['bubble' => 'Diagrama de burbujas', 'lineas' => '
            Diagrama de líneas', 'semi' => 'Diagrama semicirculo', 'circulo' => 'Diagrama circular', 'barras' => '
            Diagrama de barras', 'todo' => 'Diagrama de todo'], ['id' => 'tgrafico5']);
        ?>
        <?=
        $form->field($model, 'selectDatos5')->dropDownList(['laboral' => 'Situación laboral', 'genero' => 'Hombres
            y Mujeres en España', 'edad' => 'Edades España'], ['id' => 'tdatos5']);
        ?>
    </div>
    <div id="mando6" class="col-sm-2 col-xs-6">
        <?=
        $form->field($model, 'tipoGrafico6')->dropDownList(['lineas' => 'Diagrama de líneas', 'semi' => 'Diagrama
            semicirculo', 'circulo' => 'Diagrama circular', 'barras' => 'Diagrama de barras', 'bubble' => 'Diagrama
            de burbujas', 'todo' => 'Diagrama de todo'], ['id' => 'tgrafico6']);
        ?>
        <?=
        $form->field($model, 'selectDatos6')->dropDownList(['edad' => 'Edades España', 'genero' => 'Hombres y
            Mujeres en España', 'laboral' => 'Situación laboral'], ['id' => 'tdatos6']);
        ?>
    </div>

</div>
<div style = "text-align: center">
    <?= $form->field($model, "nombrePanel")->input("nombrePanel") ?>
    <?= Html::submitButton("Guardar Panel", ["class" => "btn btn-primary"]) ?>
</div>
```

Ilustración 27: Selección de tipo de gráficos y datos de Crear Panel.

Una vez tengamos decido como va a ser nuestro panel para guardar, nuestros datos pasarán por el modelo para comprobar que es todo correcto, es decir las reglas que hemos especificado para su creación, como son por ejemplo de que el nombre del panel este comprendido entre 3 y 50 caracteres o que incluya solo letras y números.

```php
public function rules() {
    return [['idPanel', 'integer', 'message' => 'Id incorrecto'],
            ['tipoGrafico', 'required', 'message' => 'Selecciona un grafico'],
            ['tipoGrafico2', 'required', 'message' => 'Selecciona un grafico'],
            ['tipoGrafico3', 'required', 'message' => 'Selecciona un grafico'],
            ['tipoGrafico4', 'required', 'message' => 'Selecciona un grafico'],
            ['tipoGrafico5', 'required', 'message' => 'Selecciona un grafico'],
            ['tipoGrafico6', 'required', 'message' => 'Selecciona un grafico'],
            ['selectDatos', 'required', 'message' => 'Selecciona los datos'],
            ['selectDatos2', 'required', 'message' => 'Selecciona los datos'],
            ['selectDatos3', 'required', 'message' => 'Selecciona los datos'],
            ['selectDatos4', 'required', 'message' => 'Selecciona los datos'],
            ['selectDatos5', 'required', 'message' => 'Selecciona los datos'],
            ['selectDatos6', 'required', 'message' => 'Selecciona los datos'],
            ['numGraficos', 'required', 'message' => 'Selecciona los datos'],
            ['nombrePanel', 'required', 'message' => 'Introduce un nombre para guardar el panel'],
            ['nombrePanel', 'match', 'pattern' => '/^[a-záéíóúñ0-9\s]+$/i', 'message' => 'Nombre de panel único'],
            ['nombrePanel', 'match', 'pattern' => '/^.{3,50}$/', 'message' => 'Mínimo 3 máximo 50 caracteres'],];
}
```

Ilustración 28: Reglas para el modelo de creación de un panel.

Para terminar, se iría a la parte del controlador para hacer la acción de guardado tal y como se muestra en la siguiente ilustración.

```php
public function actionSpain()
{
    $msg = null;
    $model = new FormSpain();
    $iduserlogged = Yii::$app->user->identity->id;

    if ($model->load(Yii::$app->request->post())) {
        if ($model->validate()) {
            $table = new Panels;
            $table->idUser = $iduserlogged;
            $table->nombrePanel = $model->nombrePanel;
            $table->fechaCreado = date("Y-m-d H:i:s");
            $table->numGraficos = $model->numGraficos;
            $table->grafico1 = $model->tipoGrafico;
            $table->dato1 = $model->selectDatos;
            $table->grafico2 = $model->tipoGrafico2;
            $table->dato2 = $model->selectDatos2;
            $table->grafico3 = $model->tipoGrafico3;
            $table->dato3 = $model->selectDatos3;
            $table->grafico4 = $model->tipoGrafico4;
            $table->dato4 = $model->selectDatos4;
            $table->grafico5 = $model->tipoGrafico5;
            $table->dato5 = $model->selectDatos5;
            $table->grafico6 = $model->tipoGrafico6;
            $table->dato6 = $model->selectDatos6;
            $table->departamento = "España";


            if ($table->insert()) {
                $msg = "Añadiendo a sus paneles personalizados...";
                echo "<meta http-equiv='refresh' content='2; " . Url::toRoute("site/administrarpaneles") . "'>";
            }else {
                $msg = "Ha ocurrido un error al llevar a cabo tu guardado";
            }
        }else {
            $model->getErrors();
        }
    }

    return $this->render('spain', ["model" => $model, "msg" => $msg]);
}
```

Ilustración 29: Controlador acción de guardado panel nuevo.

Como se puede observar, se comprueba antes de enviar los datos a la tabla de paneles de la base de datos que el modelo sea correcto y una vez comprobado se incluyen todos los datos para su creación, si todo es correcto nos redirigirá al apartado de "Administrar mis paneles" para ver su inserción en él.

Para destacar en este punto de crear paneles un problema que se solventó es que HighCharts cada gráfico tiene una forma concreta de incluir los datos, unos son con arrays donde le pasas los datos u otros hay que hacer una estructura de JSON para poder incluirlos.

```
function imprimirDatos(arrayDatos){
    let imprimir = [];

    for(var i = 0; i < arrayDatos.length; i++)
    {
        imprimir.push({name: arrayDatos[i].name , value: arrayDatos[i].y});
    }

    return JSON.parse(JSON.stringify(imprimir));
}
```

Ilustración 30: Creación objeto JSON para datos HighCharts.

En los gráficos que eran necesario esta estructura de datos, se creó una función que recorriera el array de los datos a imprimir y le creábamos su estructura JSON.

Así que a continuación vamos a mostrar unos ejemplos de paneles de visualización creados con distintos gráficos, cantidad de gráficos que aparecen y que datos aparecen en cada uno de ellos.

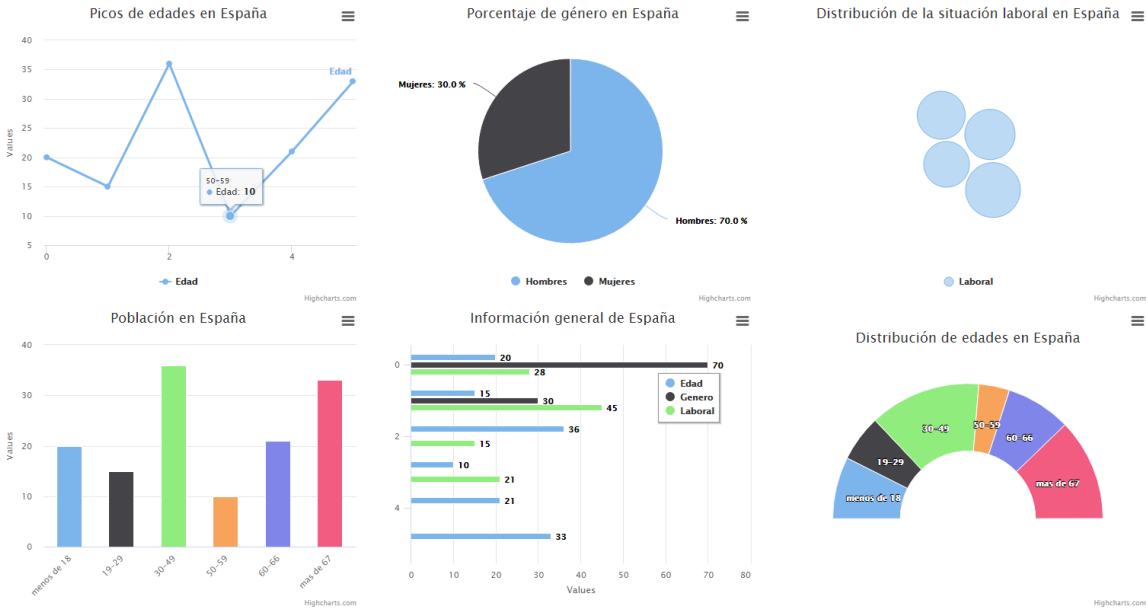

Ilustración 31: Ejemplo panel de visualización con datos de España.

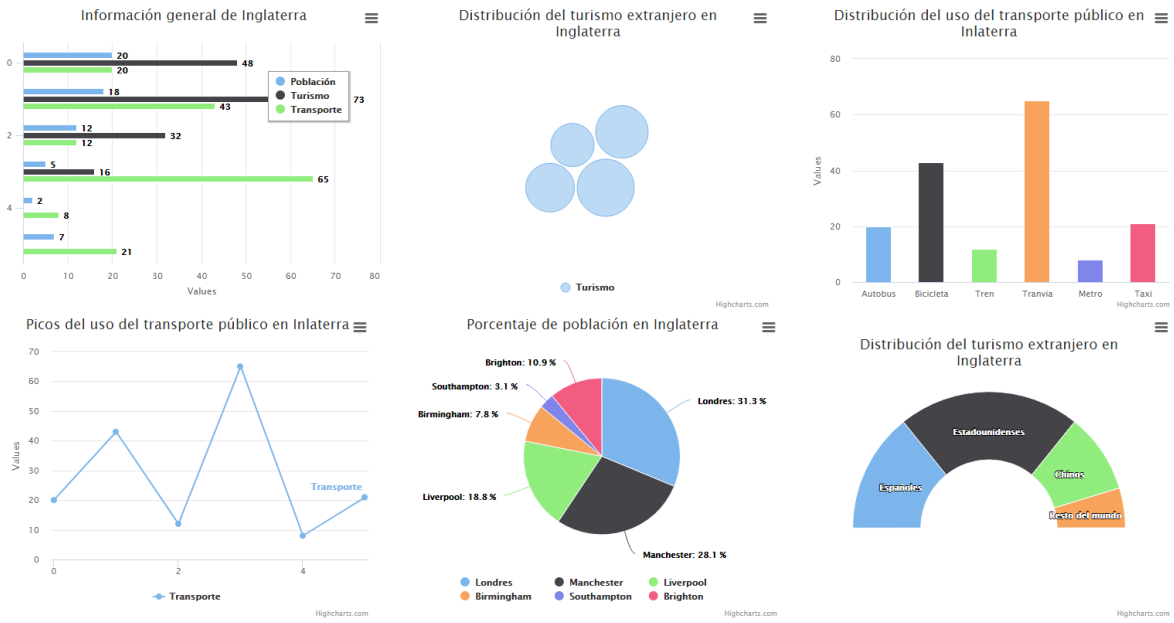

Ilustración 32: Ejemplo panel de visualización con datos de Inglaterra.

Ilustración 33: Ejemplo panel de visualización con cuatro gráficos.

6.4.2 Administrar mis paneles

Cuando se crea un panel nuevo seremos redirigidos a este apartado y se nos mostrará de la siguiente forma.

Ilustración 34: Administración de mis paneles.

Para tener esta tabla, desde la vista tenemos un bucle foreach que con el cual según el id de cada usuario traemos desde una acción del controlador todos los paneles que ha creado dicho usuario y le añadimos las acciones de editar y eliminar.

```
    <body>
            <table class="table table-striped table-bordered" style="width: 80%">
        <tr>
            <th>Nombre Panel</th>
            <th>Fecha Creación</th>
            <th>Departamento</th>
        </tr>
        <?php foreach ($model as $row): ?>
        <tr>
            <td><?= $row->nombrePanel ?></td>
            <td><?= $row->fechaCreado ?></td>
            <td><?= $row->departamento ?></td>
            <?php if($row->departamento === 'España'){?>
                    <td><a href="<?= Url::toRoute(["site/updatespain", "idPanel" => $row->idPanel]) ?>">Editar</a></td>
            <?php } ?>

            <?php if($row->departamento === 'Inglaterra'){?>
                    <td><a href="<?= Url::toRoute(["site/updateengland", "idPanel" => $row->idPanel]) ?>">Editar</a></td>
            <?php } ?>
            <td>
                <a href="#" data-toggle="modal" data-target="#idPanel<?= $row->idPanel ?>">Eliminar</a>
                    <div class="modal fade" role="dialog" aria-hidden="true" id="idPanel<?= $row->idPanel ?>">
                        <div class="modal-dialog">
                            <div class="modal-content">
                                <div class="modal-header">
                                    <button type="button" class="close" data-dismiss="modal" aria-label="Close"><span aria-hidden="
                                    true">x</span></button>
                                    <h4 class="modal-title">Eliminar Panel</h4>
                                </div>
                                <div class="modal-body">
                                    <p>¿Realmente deseas eliminar al Panel <?= $row->nombrePanel ?>?</p>
                                </div>
                                <div class="modal-footer">
                                    <?= Html::beginForm(Url::toRoute("site/deletepanel"), "POST") ?>
                                    <input type="hidden" name="idPanel" value="<?= $row->idPanel ?>">
                                    <button type="button" class="btn btn-default" data-dismiss="modal">Cerrar</button>
                                    <button type="submit" class="btn btn-primary">Eliminar</button>
                                    <?= Html::endForm() ?>
                                </div>
                            </div>
                        </div>
                    </div>
            </td>
        </tr>
    <?php endforeach ?>
    </table>
<?=
LinkPager::widget([
    "pagination" => $pages,
]);

?>
```

Ilustración 35: Vista Administrar mis paneles.

```php
public function actionAdministrarpaneles() {

    $form = new FormSearch;
    $search = null;

    if ($form->validate()) {
            $search = Yii::$app->user->identity->id;
            $table = Panels::find()
                    ->where(["like", "idUser", $search]);
            $count = clone $table;
            $pages = new Pagination([
                "pageSize" => 5,
                "totalCount" => $count->count()
            ]);
            $model = $table
                    ->offset($pages->offset)
                    ->limit($pages->limit)
                    ->all();

    }else {
        $table = Panels::find();
        $count = clone $table;
        $pages = new Pagination([
            "pageSize" => 5,
            "totalCount" => $count->count(),
        ]);
        $model = $table
                ->offset($pages->offset)
                ->limit($pages->limit)
                ->all();
    }
    return $this->render("administrarpaneles", ["model" => $model, "form" => $form, "search" => $search, "pages" => $pages]);

}
```

Ilustración 36: Acción Administrar mis paneles.

Como podemos ver hacemos una consulta con el id del usuario conectado en ese momento a la tabla paneles de la base de datos y nos traemos todos los datos para montarlo en la tabla que mostramos en la ilustración anterior, además tenemos la condición de que si vienen más de cinco gráficos incluiremos una paginación para que sea todo más amigable de ver.

Tras esto vamos a explicar como es el proceso de poder editar un panel ya creado, el proceso es parecido a la hora de crearlo, pero con la peculiaridad de que en la acción del controlador tenemos que traer la información de ese panel y montarla en el modelo para que se inserte en la vista.

```php
if (Yii::$app->request->get("idPanel")) {
    $idPanel = Html::encode($_GET["idPanel"]);
    if ((int) $idPanel) {
        $table = Panels::findOne($idPanel);
        if ($table) {
            $model->idPanel = $table->idPanel;
            $model->nombrePanel = $table->nombrePanel;
            $model->numGraficos = $table->numGraficos;
            $model->tipoGrafico = $table->grafico1;
            $model->selectDatos = $table->dato1;
            $model->tipoGrafico2 = $table->grafico2;
            $model->selectDatos2 = $table->dato2;
            $model->tipoGrafico3 = $table->grafico3;
            $model->selectDatos3 = $table->dato3;
            $model->tipoGrafico4 = $table->grafico4;
            $model->selectDatos4 = $table->dato4;
            $model->tipoGrafico5 = $table->grafico5;
            $model->selectDatos5 = $table->dato5;
            $model->tipoGrafico6 = $table->grafico6;
            $model->selectDatos6 = $table->dato6;
        } else {
            return $this->redirect(["site/index"]);
        }
    } else {
        return $this->redirect(["site/index"]);
    }
} else {
    return $this->redirect(["site/index"]);
}
return $this->render("updatespain", ["model" => $model, "msg" => $msg]);
```

Ilustración 37: Controlador acción actualizar panel.

Como se puede observar mediante el "idPanel" que es la Primary Key de nuestro panel obtenemos todos sus datos mediante la consulta a la tabla paneles de la base de datos, con esto enviamos al modelo todos los campos que tiene dicho panel y los insertaríamos en la vista para su modelado, es decir el caso inverso a la creación.

Cuando ya se hayan actualizado los campos a desear, como por ejemplo añadir más gráficos o quitar alguno, cambiar el nombre de ese panel, cambiar tipos de gráficos o sus datos, se enviarán estos nuevos campos actualizados a la acción del controlador que realiza este proceso, no antes sin pasar por el modelo para ver que estén correctos todos los campos, para su inserción en la base de datos.

```php
public function actionUpdatespain() {
    $model = new FormSpain;
    $msg = null;

    if ($model->load(Yii::$app->request->post())) {
        if ($model->validate()) {
            $table = Panels::findOne($model->idPanel);
            if ($table) {
                $table->idPanel = $model->idPanel;
                $table->nombrePanel = $model->nombrePanel;
                $table->fechaCreado = date("Y-m-d H:i:s");
                $table->numGraficos = $model->numGraficos;
                $table->grafico1 = $model->tipoGrafico;
                $table->dato1 = $model->selectDatos;
                $table->grafico2 = $model->tipoGrafico2;
                $table->dato2 = $model->selectDatos2;
                $table->grafico3 = $model->tipoGrafico3;
                $table->dato3 = $model->selectDatos3;
                $table->grafico4 = $model->tipoGrafico4;
                $table->dato4 = $model->selectDatos4;
                $table->grafico5 = $model->tipoGrafico5;
                $table->dato5 = $model->selectDatos5;
                $table->grafico6 = $model->tipoGrafico6;
                $table->dato6 = $model->selectDatos6;

                if ($table->update()) {
                    $msg = "El panel ha sido actualizado correctamente";
                } else {
                    $msg = "El panel no ha podido ser actualizado";
                }
            } else {
                $msg = "El panel seleccionado no ha sido encontrado";
            }
        } else {
            $model->getErrors();
        }
    }
```

Ilustración 38: Controlador acción actualizar panel.

Por último, para eliminar un panel cualquiera, al pulsar sobre el botón de eliminar, saltará una ventana emergente para que se confirme dicha acción, ya que es un proceso irreversible para tener una mayor seguridad en dicho proceso.

Ilustración 39: Ventana emergente eliminar panel.

Para realizar este proceso, tenemos que crear una acción en el controlador donde buscaremos en la base de datos mediante la Primary Key del panel y al ser un atributo único lo eliminamos con la función "DeleteAll".

```php
public function actionDeletepanel()
{
    if (Yii::$app->request->post()) {
        $idPanel = Html::encode($_POST["idPanel"]);
        if ((int) $idPanel) {
            if (Panels::deleteAll("idPanel=:idPanel", [":idPanel" => $idPanel])) {
                echo "Panel $idPanel eliminado con éxito, redireccionando ...";
                echo "<meta http-equiv='refresh' content='3; " . Url::toRoute("site/administrarpaneles") . "'>";
            } else {
                echo "Ha ocurrido un error al eliminar el panel, redireccionando ...";
                echo "<meta http-equiv='refresh' content='3; " . Url::toRoute("site/administrarpaneles") . "'>";
            }
        } else {
            echo "Ha ocurrido un error al eliminar el panel, redireccionando ...";
            echo "<meta http-equiv='refresh' content='3; " . Url::toRoute("site/administrarpaneles") . "'>";
        }
    } else {
        return $this->redirect(["site/index"]);
    }
}
```

Ilustración 40: Controlador acción eliminar panel.

6.4.3 Mi perfil

En este punto tenemos la información referente a cada usuario, la cual es totalmente accesible para cada uno y con la posibilidad de actualizar tanto sus datos personales como cambiar la contraseña.

Para poder realizar este proceso hemos creado una acción en el controlador con la cual obtenemos los datos a mostrar en la vista y una vez se haya realizado los cambios en alguno de los campos del formulario, estos son revisados por el modelo para luego ser pasado de nuevo por el controlador para su actualización en la base de datos.

```php
public function actionProfile()
{
    $msg = "";
    $model = new FormUser();
    $iduserlogged = Yii::$app->user->identity->id;
    $table = Users::findOne($iduserlogged);

    if ($model->load(Yii::$app->request->post())) {
        if ($model->validate()) {
            if ($table) {
                $table->id = $model->id;
                $table->nombre = $model->nombre;
                $table->email = $model->email;
                $table->password = utf8_encode(Yii::$app->getSecurity()->encryptByKey($model->password, 'key123'));
                $table->telefono = $model->telefono;
                $table->departamento = $model->departamento;
                if ($table->update()) {
                    $msg = "El perfil ha sido actualizado correctamente";
                } else {
                    $msg = "El perfil no ha podido ser actualizado";
                }
            } else {
                $msg = "ERROR: No se ha encontrado el perfil del usuario con id: " . $iduserlogged;
            }
        } else {
            $model->getErrors();
        }
    }

    if ($table) {
        $model->id = $table->id;
        $model->nombre = $table->nombre;
        $model->email = $table->email;
        $model->password = Yii::$app->getSecurity()->decryptByKey(utf8_decode($table->password), 'key123');
        $model->telefono = $table->telefono;
        $model->departamento = $table->departamento;
        $model->esadmin = $table->esadmin;
    } else {
        return $this->redirect(['site/index']);
    }
    return $this->render('profile', ["model" => $model, "msg" => $msg]);
}
```

Ilustración 41: Controlador acción actualizar datos perfil.

Para darle algo más de seguridad en el campo de la contraseña ya que es un campo más sensible que el resto, he optado por usar la función de Yii2 que es "EncryptByKey", que con la cual pasamos nuestro parámetro a encriptar y una key secreta que nosotros establezcamos para realizar el proceso tanto de encriptado como el de desencriptado.

```php
public function rules() {
    return [
        [['nombre', 'email'], 'required', 'message' => 'Campo requerido, no puede dejarse en blanco'],
        // email has to be a valid email address
        ['email', 'email', 'message' => 'Dirección de correo no válida'],
        ['id', 'integer', 'message' => 'Id incorrecto'],
        ['nombre', 'match', 'pattern' => '/^[a-záéíóúñ\s]+$/i', 'message' => 'Sólo se aceptan letras'],
        ['nombre', 'match', 'pattern' => '/^.{3,50}$/', 'message' => 'Mínimo 3 máximo 50 caracteres'],
        ['password', 'match', 'pattern' => '/^.{8,20}$/', 'message' => 'Mínimo 8 máximo 20 caracteres'],
        ['telefono', 'number', 'message' => 'Campo numérico'],
        ['departamento', 'match', 'pattern' => "/^[^$%&|<>#]*$/", 'message' => 'No se permiten los siguientes caracteres en este
            campo: $ % & | < > # '],
    ];
}
```

Ilustración 42: Reglas modelo del formulario perfil.

```php
<h1>Perfil de <?= Html::encode($model["nombre"]) ?></h1>

<h3><?= $msg ?></h3>

<?php
$form = ActiveForm::begin([
            "method" => "post",
            'enableClientValidation' => true,
        ]);
?>

<?= $form->field($model, "id")->input("hidden")->label(false) ?>

<div class="form-group">
    <?= $form->field($model, "nombre")->input("text") ?>
</div>

<div class="form-group">
    <?= $form->field($model, "email")->input("text") ?>
</div>

<div class="form-group">
    <?= $form->field($model, "password")->input("password")->input("password", ["id" => 'password']) ?>
    <div style="float: right;"><input type="checkbox" onclick="showPass()"> Mostrar contraseña</div>
</div>
<br>
<div class="form-group">
    <?= $form->field($model, "telefono") ?>
</div>
<?php if (!$model['esadmin']) {
    ?>
    <div class="form-group">
        <?= $form->field($model, 'departamento')->dropDownList(['España' => 'España', 'Inglaterra' => 'Inglaterra', 'Alemania' => '
            Alemania'], ['disabled' => true]); ?>
        <?= $form->field($model, "departamento")->input("hidden")->label(false) ?>
    </div>
    <?php
}
?>


<?= Html::submitButton("Actualizar", ["class" => "btn btn-primary"]) ?>

<?php $form->end() ?>
```

Ilustración 43: Código vista "Mi perfil".

6.4.4 Administrar usuarios

Para el apartado de administración de usuarios hemos seguido un modelo parecido a la hora de administrar los paneles de cada usuario, salvo que en este caso al tener un volumen mayor de usuarios registrados que de paneles creados, se ha optado por añadir un buscador de usuarios para que sea más fácil la búsqueda de algún usuario en concreto.

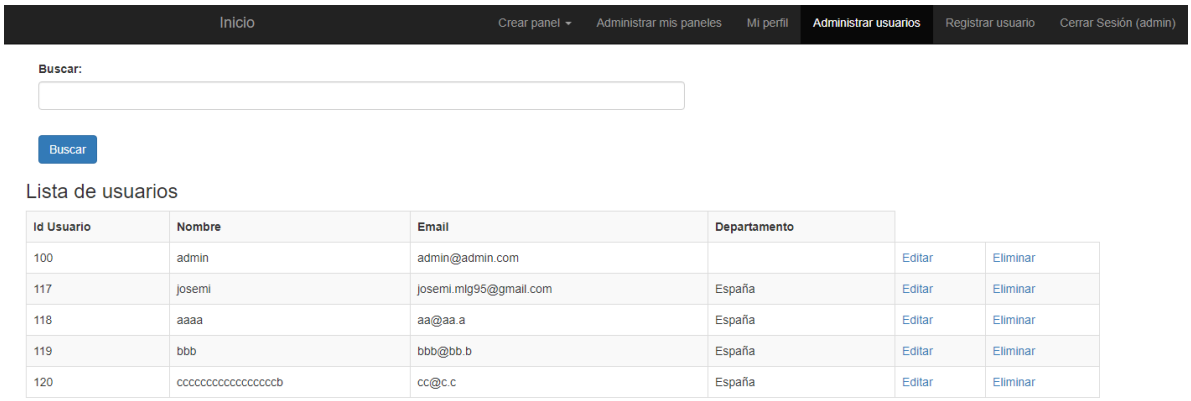

Ilustración 44: Interfaz "Administrar usuarios".

Como la parte de creación de la tabla de usuarios registrado es parecida a la de paneles de visualización, vamos a explicar como se realiza el proceso de búsqueda.

```php
<?php
$f = ActiveForm::begin([
            "method" => "get",
            "action" => Url::toRoute("site/administracion"),
            "enableClientValidation" => true,
        ]);
?>
<div class="row">
    <div class="col-sm-12">
        <div class="form-group col-sm-6">
<?= $f->field($form, "q")->input("search") ?>
        </div>
        <div class="col-sm-12 col-xs-12">
<?= Html::submitButton("Buscar", ["class" => "btn btn-primary"]) ?>
        </div>
    </div>
</div>

<?php $f->end() ?>
```

Ilustración 45: Código vista administrar usuarios.

```php
class FormSearch extends model {

    public $q;

    public function rules() {
        return [["q", "match", "pattern" => "/^[0-9a-záéíóúñ\s]+$/i", "message" => "Solo se aceptan letras y números"]
        ];
    }

    public function attributeLabels() {
        return ["q" => "Buscar:"];
    }

}
```

Ilustración 46: Formulario de búsqueda administrar usuarios.

```php
public function actionAdministracion()
{
    $form = new FormSearch;
    $search = null;

    if ($form->load(Yii::$app->request->get())) {
        if ($form->validate()) {
            $search = Html::encode($form->q);
            $table = Users::find()
                    ->where(["like", "id", $search])
                    ->orWhere(["like", "email", $search])
                    ->orWhere(["like", "nombre", $search])
                    ->orWhere(["like", "departamento", $search]);
            $count = clone $table;
            $pages = new Pagination([
                "pageSize" => 5,
                "totalCount" => $count->count()
            ]);
            $model = $table
                    ->offset($pages->offset)
                    ->limit($pages->limit)
                    ->all();
        } else {
            $form->getErrors();
        }
```

Ilustración 47: Acción controlador administrar usuarios.

Como se puede observar en las ilustraciones anteriores lo que hemos hecho es crear en la vista una sección para la búsqueda, en la cual guardamos los parámetros insertados en el campo "Buscar" para comprobar mediante el modelo que cumple las reglas que le hemos puesto (Ilustración 41) y tras ello filtrar en la acción del controlador a la hora de realizar la búsqueda en la base de datos.

Los campos con los que se puede realizar una búsqueda filtrada son con el id del usuario, su email, su nombre de usuario y su departamento.

6.4.5 Registrar usuario

Para poder incluir los usuarios registrados en el apartado de administrar los usuarios antes se ha tenido que realizar el proceso de registrado. En este caso y como ya comentamos en los apartados iniciales de este capítulo de la memoria, se optó por usar un modelo de envío de solicitudes mediante un formulario "Solicitar Registro".

Por lo que el usuario administrador al ver las solicitudes que va recibiendo, este añade a estos usuarios y una vez realizado este proceso, a la hora de guardar, se envía un correo al usuario registrado con sus credenciales de acceso a la plataforma.

```
<h1>Registro de usuario</h1>

<?php
$form = ActiveForm::begin([
            'method' => 'post',
            'id' => 'formulario',
        ]);
?>

<div class="form-group">
<?= $form->field($model, "nombre")->input("text") ?>
</div>

<div class="form-group">
<?= $form->field($model, 'departamento')->dropDownList(['España' => 'España', 'Inglaterra' => 'Inglaterra', 'Alemania' => 'Alemania'],
    ['prompt' => 'Selecciona uno']); ?>
</div>

<div class="form-group">
<?= $form->field($model, "email")->input("email") ?>
</div>

<div class="form-group">
<?= $form->field($model, "telefono")->input("text") ?>
</div>

<div class="form-group">
<?= $form->field($model, "password")->input("password") ?>
</div>

<div class="form-group">
<?= $form->field($model, "password_repeat")->input("password") ?>
</div>

<?= Html::submitButton("Registrar", ["class" => "btn btn-primary"]) ?>

<?php $form->end() ?>
```

Ilustración 48: Código vista registrar usuarios.

```
public function rules()
{
    return [
        [['nombre', 'email', 'departamento','password', 'password_repeat'], 'required', 'message' => 'Campo requerido'],
        ['nombre', 'match', 'pattern' => '/^[a-záéíóúñ\s]+$/i', 'message' => 'Sólo se aceptan letras'],
        ['nombre', 'match', 'pattern' => '/^.{3,50}$/', 'message' => 'Mínimo 3 máximo 50 caracteres'],
        ['nombre', 'nombre_existe'],
        ['email', 'match', 'pattern' => "/^.{5,80}$/", 'message' => 'Mínimo 5 y máximo 80 caracteres'],
        ['email', 'email', 'message' => 'Formato no válido'],
        ['email', 'email_existe'],
        ['password', 'match', 'pattern' => "/^.{8,16}$/", 'message' => 'Mínimo 6 y máximo 16 caracteres'],
        ['password_repeat', 'compare', 'compareAttribute' => 'password', 'message' => 'Las contraseñas no coinciden'],
        ['telefono', 'number', 'message' => 'Campo numérico'],
        ['departamento', 'required', 'message' => 'Selecciona un departamento']
    ];
}
```

Ilustración 49: Código reglas del formulario registrar usuarios.

```php
public function email_existe($attribute, $params)
{
    $table = Users::find()->where("email=:email", [":email" => $this->email]);

    if ($table->count() == 1)
    {
        $this->addError($attribute, "Este email ya está registrado");
    }
}

public function nombre_existe($attribute, $params)
{
    $table = Users::find()->where("nombre=:nombre", [":nombre" => $this->nombre]);

    if ($table->count() == 1)
    {
        $this->addError($attribute, "Este nombre de usuario ya esta en uso");
    }
}
```

Ilustración 50: Comprobar existencia de usuario.

```php
public function actionRegister()
{
    $model = new FormRegister;
    $msg = null;

    if ($model->load(Yii::$app->request->post()) && Yii::$app->request->isAjax) {
        Yii::$app->response->format = Response::FORMAT_JSON;
        return ActiveForm::validate($model);
    }

    if ($model->load(Yii::$app->request->post())) {
        if ($model->validate()) {
            $table = new Users;
            $table->nombre = $model->nombre;
            $table->email = $model->email;
            $table->departamento = $model->departamento;
            $table->telefono = $model->telefono;

            $table->password = utf8_encode(Yii::$app->getSecurity()->encryptByKey($model->password, 'key123'));

            if ($table->insert()) {
                $user = $table->find()->where(["email" => $model->email])->one();
                $subject = "Bienvenido al portal web";
                $body = "<h1>Datos de inicio de sesión</h1><br></br>"
                        . "<p>Usuario: " . $model->nombre . "</p>"
                        . "<p>Contraseña: " . $model->password . "</p>"
                        . "<p>Departamento: " . $model->departamento . "</p>";
                $content = "<p>Usuario: " . $model->nombre . "</p>";
                $content .= "<p>Contraseña: " . $model->password . "</p>";

                Yii::$app->mailer->compose("@app/mail/layouts/html", ["content" => $content])
                        ->setTo($user->email)
                        ->setFrom([Yii::$app->params["adminEmail"] => Yii::$app->params["title"]])
                        ->setSubject($subject)
                        ->setHtmlBody($body)
                        ->send();

                $model->nombre = null;
                $model->email = null;
                $model->password = null;
                $model->password_repeat = null;

                $msg = "Enhorabuena, usuario registrado y enviado el correo con sus datos de inicio";
            } else {
                $msg = "Ha ocurrido un error al llevar a cabo tu registro";
            }
        } else {
            $model->getErrors();
        }
    }

    return $this->render("register", ["model" => $model, "msg" => $msg]);
```

Ilustración 51: Acción controlador registrar usuario.

Seguimos con el modelo-vista-controlador (MVC) y en este caso tenemos que, una vez recogidos los campos del formulario de la vista de registro, comprobamos la existencia de si dicho usuario, mediante una consulta en la tabla de usuarios que tenemos en la base de datos, está registrado no seguirá el proceso y será avisado al administrador.

Tras este proceso, en el apartado del controlador tenemos que una vez que haya sido insertado en la tabla de usuarios se crea un modelo de correo con los campos que sean necesarios enviar al usuario para su acceso en la plataforma y mediante Yii2 realizamos el proceso de envío de correos electrónicos.

6.5 Pantalla principal usuario normal

Por último, los usuarios están categorizados según el departamento al que pertenecen por lo que, si no son administradores tienen restricciones a ciertas funcionalidades que hemos visto en el subcapítulo anterior.

Ilustración 52: Interfaz usuario no administrador.

Como se puede observar, tenemos que estos usuarios tienen la posibilidad de crear sus paneles según los datos que tengan accesibles, administrarlos y ver y modificar los datos referentes a su perfil.

Una de las cosas interesantes de poder saber como mostrar que datos tienen disponibles o no cada usuario es desde la vista traernos a que departamento pertenecen y visibilizar solo aquellos a los que tengan acceso, tal y como se muestra en la siguiente ilustración.

```
[ 'label' => 'Crear panel',
    'items' => [['label' => 'España', 'url' => ['/site/spain'], 'visible' => (Yii::$app->user->
        identity->nombre == 'admin' || Yii::$app->user->identity->departamento == 'España')],
        '<li class="divider"></li>',
        ['label' => 'Inglaterra', 'url' => ['/site/england'], 'visible' => (Yii::$app->user->
            identity->nombre == 'admin' || Yii::$app->user->identity->departamento == 'Inglaterra'
            )],
        '<li class="divider"></li>',
    ],
]
),
```

Ilustración 53: Código control de "Crear panel".

7

Conclusiones y trabajo futuro

7.1 Conclusiones

La elaboración de este proyecto ha supuesto un amplio reto de aprendizaje en diferentes aspectos del ámbito de la informática, todo este proceso se puede desglosar en las siguientes conclusiones:

- En primer lugar, en la selección de tecnologías que se iban a usar para la elaboración de este proyecto, como lenguajes de programación, frameworks, programas informáticos, etc., se ha buscado las que a día de hoy son las más usadas en el ámbito de la creación de aplicaciones web por lo que ese era uno de los objetivos de usar todas estas herramientas, estar lo más actualizado a la demanda de hoy día.

- Por otra parte, tenemos el estudio realizado para la elección de que herramienta se iba a encargar de crear los paneles de visualización de la aplicación. Este punto es uno muy importante por este motivo, por lo que se ha buscado una mezcla entre herramientas muy demandadas hoy en día como es el caso de Power BI o herramientas de no tan conocidas como es el caso de la seleccionada en dicho estudio, HighCharts.

- Un último punto de conclusiones, pero es referente a la forma interna de seguir el proyecto, es decir, la metodología de trabajo y es que en referencia a esto se ha intentado seguir como se haría en empresas de este sector. Por lo que como se menciona al inicio de esta memoria se ha seguido una metodología en cascada, la cual se ha seguido correctamente.

Para terminar, estos puntos concluyen en definitiva que se ha buscado una herramienta que sea por y para el usuario que le vaya a usar, es decir, que no sea de muy complejo uso como puede ser con otras herramientas parecidas a esta, ya que normalmente estas están mucho más sobrecargadas de funcionalidades de todo tipo y que no son normalmente de fácil uso.

Este último aspecto se ha solventado con una interfaz amigable gracias a la herramienta de creación de paneles seleccionada y a como está estructurada toda la aplicación y con esto se consigue tener un acceso a toda la información necesaria de cada uno o la distribución de las funcionalidades necesarias de cada usuario de una forma rápida y de sencillo manejo.

7.2 Trabajo futuro

Con este proyecto se ha buscado asentar las bases para tener una aplicación web funcional para poder tener los paneles de visualización de los datos internos de una empresa, mediante el estudio profundo sobre que herramienta finalmente usar para crear dichos paneles y todo le referente a la gestión de usuarios, por lo que no ha dado tiempo a poder realizar otras funcionalidades interesantes y las cuales pueden quedar para un trabajo futuro, las cuales podrían ser las siguientes:

- Se podría mejorar el aspecto de la seguridad en las contraseñas ya que no es muy complejo de la manera en la que está implementado actualmente.

- Enlazando con el punto anterior, se podría establecer un protocolo de transferencia segura de datos como puede ser HTTPS.

- Otra posible mejora sería la posibilidad de crear los paneles de visualización con datos de proveniente de otras plataformas que no sea MySQL, como puede ser provenientes de hojas de Excel, CSV o con conexiones con herramientas como Google Analytics o Adobe Analytics.

- Se podría también cambiar que el registro de usuario para que sea más automático en vez de con solicitud de alta por correo electrónico, esto fue una decisión personal como ya expliqué anteriormente para usar funcionalidades del framework Yii2.

Referencias

[1] Manual Yii2: https://www.yiiframework.com/doc/guide/2.0/es
[2] Manual PHP: https://www.php.net/manual/es/index.php
[3] Manual JavaScript: https://desarrolloweb.com/manuales/manual-javascript.html
[4] Manual MySQL: https://dev.mysql.com/doc/refman/8.0/en/language-structure.html
[5] Instalación XAMPP: http://www.mclibre.org/consultar/php/otros/xampp-instalacion-windows.html
[6] Documentación Power BI: https://docs.microsoft.com/es-es/power-bi/power-bi-overview
[7] Documentación Tableau Public: https://public.tableau.com/en-us/s/resources
[8] Documentación Visualize.js: https://community.jaspersoft.com/documentation/tibco-jasperreports-server-visualizejs-guide/v71/api-reference-visualizejs
[9] Documentación HighCharts: https://www.highcharts.com/docs/
[10] MVC Yii2: https://www.yiiframework.com/doc/guide/1.1/es/basics.mvc
[11] Manual GitHub: https://education.github.com/git-cheat-sheet-education.pdf

Apéndice A

Manual de Instalación

En este manual de instalación se explicará paso por paso como ha sido el proceso que he utilizado para poder trabajar con el proyecto de manera local antes de subir los avances al servidor proporcionado. Esto como se explica en la documentación anterior se realiza mediante un servidor apache y el SGBD PhpMyAdmin, utilizando la herramienta XAMPP.

1. Acceder a la pagina web de XAMPP https://www.apachefriends.org/es/index.html
2. Seleccionamos la versión que más acorde sea a nuestro sistema operativo.
3. Cuando lo tengamos instalado, iniciamos XAMPP y deberemos pulsar el botón de "Start" tanto en "Apache" como en "MySQL".
4. Una vez iniciado para comprobar que funciona correctamente, accederemos mediante un navegador web a la URL http://localhost/xampp/
5. Para configurar la base de datos deberemos acceder a PhpMyAdmin mediante la siguiente ruta http://localhost/phpmyadmin/ que aquí crearemos la base de datos a usar, tablas, sentencias MySQL, etc.
6. Tras esto, crearemos nuestro proyecto en el directorio C:\xampp\htdocs que es donde se guardan todos los proyectos creados en XAMPP.
7. Para acabar, quedaría por relacionar nuestra aplicación con la base de datos creada y para ello deberemos dirigirnos a la siguiente ruta: C:\xampp\htdocs\proyectoFinDeGrado\config y modificar el archivo db.php que quedaría de la siguiente forma.

```
return [
    'class' => 'yii\db\Connection',
    'dsn' => 'mysql:host=localhost;dbname=databasetfg',
    'username' => 'root',
    'password' => '',
    'charset' => 'utf8',
```

8. Con esto ya estaría listo y comprobaríamos el correcto funcionamiento desde la siguiente URL http://localhost/proyectoFinDeGrado/web/